University of Massachusetts Dartmouth
Department of Electrical and Computer Engineering

ECE 320 Discrete-time Linear Systems

Fall 2013

Matlab **Project 3 Solution**

Project 7.1: Aliasing due to undersampling

This project explores how sampling continuous-time signals of different frequencies results in different discrete-time signals, and when you see aliasing in the signals.

(a) The commands to define this signal are

```
Omega0 = 2*pi*1000;
T = 1/8192;
n = [0:8191];
t = n*T;
x = sin(Omega0*t);
```

(b) The following commands plot the first fifty samples of the cosine, as shown in Figure 1.

```
figure(1)
clf
subplot(211)
stem(n(1:50),x(1:50))
xlabel('Time (samples)')
ylabel('x[n]')
subplot(212)
plot(t(1:50),x(1:50))
xlabel('Time (seconds)')
ylabel('x(t)')
```

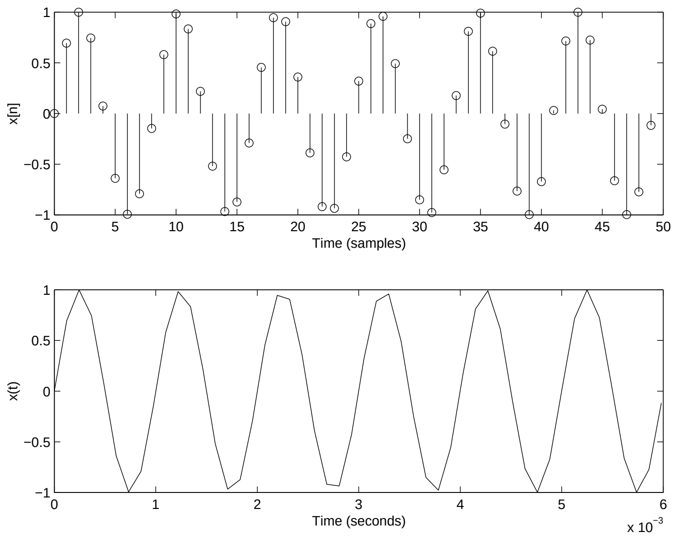

Figure 1: `plot` and `stem` for the cosine signal

(c) The reconstructed signal should have two spikes, one each at ±1000 Hz. The commands below compute this spectrum using `ctfts`, then plot the spectrum shown in Figure 2. The spikes are at the frequencies expected.

```
[X,f] = ctfts(x,T);
figure(2)
clf
subplot(211)
plot(f,abs(X))
xlabel('Frequency (Hz)')
ylabel('|X(j\omega)|')
```

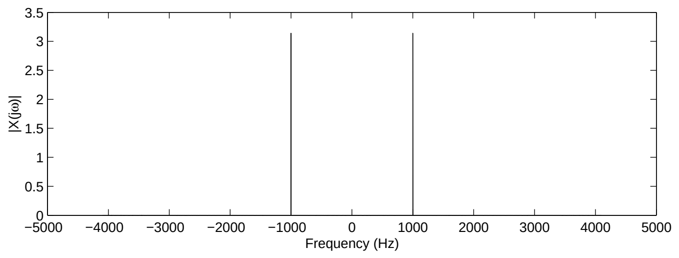

Figure 2: Magnitude spectrum $|X(f)|$ for part (c).

(d) Repeating the previous parts for the two new frequencies produces the plots shown in Figures 3 and 4. As expected, the spikes in the magnitude of the Fourier transform move to ±1500 Hz and ±2000 Hz. I've omitted the commands since they are essentially the same as above.

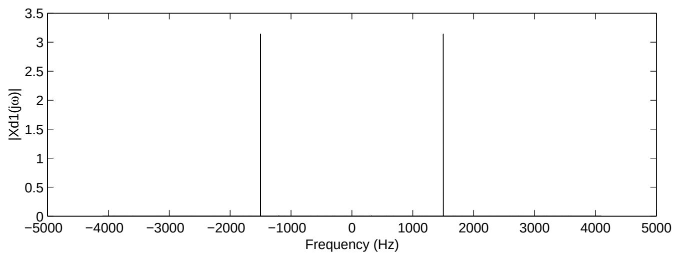

Figure 3: Magnitude spectrum $|X(f)|$ for part (d) for $2\pi(1500)$.

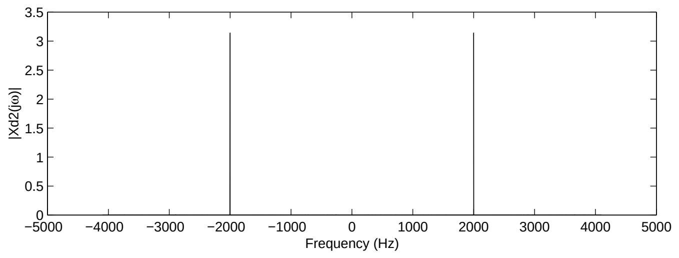

Figure 4: Magnitude spectrum $|X(f)|$ for part (d) for $2\pi(2000)$..

(e) The following commands save the two sampled signals into `.wav` files. Listening to them, I heard a tone which increased in frequency for each successive signal.

```
wavwrite(xd1,1/T,16,'Proj71xd1.wav');
wavwrite(xd2,1/T,16,'Proj71xd2.wav');
```

(f) Repeating the commands above with $\Omega_0 = 2\pi(3500), 2\pi(4000), 2\pi(4500), 2\pi(5000)$, and $2\pi(5500)$ gave a sequence of 5 more tones. The tones increased in pitch for the first two, but then started decreasing in pitch as Ω_0 was $2\pi(4500)$ and up. This makes sense because we have $T = 1/8192$ s, so the sampling frequency $\Omega_s = 2\pi(8192)$, or 8192 Hz. We expect any frequency above $\Omega_s/2 = 2\pi(4096)$ or 4096 Hz to alias. As we use the larger values of $2\pi(4500)$, $2\pi(5000)$ and $2\pi(5500)$, the tone we hear is decreasing in pitch as it aliases to lower and lower frequencies.

(g) This section deals with chirp signals, where the frequency increases linearly in time. The following commands define samples of $x(t)$ as required in the problem:

```
Omega0 = 2*pi*3000;
beta = 2*pi*2000;
xg = sin(Omega0*t+0.5*beta*t.^2);
```

(h) When I listened to the chirp, the frequency initially rose, then started to go down again at some point. This is because the frequency $\Omega_0 + \beta t$ eventually goes above half the sampling frequency, and then starts aliasing downward to lower frequencies.

(i) The chirp sounds like it starts decreasing in frequency a little more than half way through the 1 second sound. Based on the equation for instantaneous frequency, we expect the signal to sound like it is decreasing in frequency when the frequency exceeds half of the sampling frequency. This occurs when

$$\begin{aligned} \Omega_0 + \beta t &= \Omega_s/2 \\ 2\pi(3000) + 2\pi(2000)t &= 2\pi(8192)/2 \\ 3000 + 2000t &= 4096 \\ t &= (4096 - 3000)/2000 = 0.548 \text{ sec} \end{aligned}$$

This is consistent with what I heard with the pitch sounding like it reversed direction a bit more than half way through the signal.

(j) To extend the chirp, we first redefine `n` and `t` so that they are 10 times longer, using the same T, and then define a new `x`

```
n = [0:(81920-1)];
t = n*T;
xj = sin(Omega0*t+0.5*beta*t.^2);
wavwrite(xj,1/T,16,'Proj71xj.wav');
```

When I listen to this signal, it rises in pitch, then falls, then rises, and so on. Each change of direction in the pitch indicates that the instantaneous frequency has passed another multiple of $\Omega_s/2$. The sound will have very low frequency when the instantaneous frequency falls on a multiple of Ω_s, when it will alias to $\Omega = 0$ in the reconstructed signal. To determine when this happens

$$\begin{aligned} \Omega_0 + \beta t &= \Omega_s/2 \\ 2\pi(3000) + 2\pi(2000)t &= 2\pi(8192)m \text{ for integer } m \\ 3000 + 2000t &= 8192m \\ t &= (8192m - 3000)/2000 = 2.60, 6.69 \text{ sec} \end{aligned}$$

Again, these predicted times are consistent with when I hear the lowest frequencies in the sound.

New Project: Speech Scrambler

(a) Listening the speech, it is definitely scrambled. The rhythm sounds generally like speech, but it is completely unintelligible what is being said.

(b) Figure 5 shows the spectrum of the scrambled speech plotted on a dB scale. The frequencies have been converted to plot in kHz using $f = (1/2\pi)\omega = (1/2\pi)(\Omega/T)$ then scaled by 10^3. From zooming in on the plot, it appears the speech signal is contained in the regions and $|f| \leq 0.7$kHz and 3.5 kHz $\leq |f| \leq 5$ kHz. The commands below generated the figure

```
% part (a)
[x,fs] = wavread('group0.wav');

% part (b)
% find FFT size which is next largest power of 2 from signal length
Nfft = 2^nextpow2(length(x));

% find the frequencies for the fft in kHz
% assumes that we will use fftshift
X = fft(x,Nfft);
omega = ((0:(Nfft-1))*(2*pi/Nfft))-pi;
f = (omega/(2*pi))*10;
figure(1)
clf
subplot(311)
plot(f,20*log10(abs(X)));
axis([-5 5 -50 50])
grid
title('Spectrum of scrambled speech');
xlabel('Frequency (kHz)');
ylabel('|X(f)| (dB)');
```

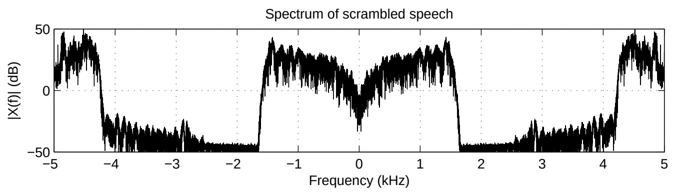

Figure 5: Spectrum of scramble speech

(c) Figure 6 shows the sequence of spectra guessed from the partial block diagram of the system that we have. Using Fourier properties, we can find that $X_c(j\omega)$ will be a triangle shaped spectrum that covers the width $\omega = \pm 2\pi(4000)$, or $|f| \leq 4$kHz. Because the signals are all real and the spectra are symmetric, the frequencies are in general only labeled for the positive frequencies to prevent the graphs from getting too cluttered. Also, the amplitude is in general not important in this sytem, so is not labeled. It is the frequencies we need to keep careful track of.

Making this spectrum required several assumptions:

- There are no gaps in spectrum below f_c. This implies that the first box on the top branch is LPF(f_1) so that no spectrum falls between the cutoff on the top branch LPF and the lower edge of the BPF on the bottom branch.

- The total bandwidth of the speech signal, even after scrambling, is $2f_c$. This is assuming that the system keeps all frequencies $|f| \leq f_c$ in one of the two branches. For the scrambled spectrum in Figure 5, this implies that $f_c \approx 2200$ Hz.
- To find the modulation frequency for the top branch, we need to look at Fig. 5 around 5 kHz. Since this modulation is followed by a BPF with upper edge 5 kHz, we know that the center of each copy in $V_1(f)$ cannot be above 5 kHz or we would lose some of the speech spectrum. If the modulation frequency was below 5 kHz, we'd see parts of the spectrum duplicated around $\pm$5kHz, rather than the two edges fitting together perfectly. (Think about how it would look if the houses in $V_1(f)$ were centered at 4.5 kHz rather than 5 kHz.) Combining all of this information, we make an informed guess that the top branch modulation signal is $2\cos(2\pi(5000)t)$.
- Looking at $V_2(f)$ in the bottom branch, we know that we only need to keep the lower frequency copies of the triangles. Also, we know that we don't want the higher frequency triangles to interfere with the house-shaped spectra in $X_1(f)$ from the top branch. Putting this together suggests that the third box on the bottom branch is a LPF with cutoff $f_c - f_1 + \Delta f$

(d) Figure 7 shows our proposed design for the descrambling system. First, we separate the signal into the high and low frequencies on either side of $\Omega = \pi/2$. The high frequencies need to be shifted from $\Omega = \pi$ back to $\Omega = 0$ where they started as the house shaped spectrum. This is done by multiplying by $\cos(\pi n)$. The small trapezoids around $\Omega = 2\pi(\Delta f)T$ need to be shifted both left and right in Ω by $2\pi(f_c + \Delta f)T$ to undo the shifts in the bottom branch of the scrambler. Once we've done this, we only want to keep the lower frequency half of each pair, so we apply a LPF with $\Omega_c = 2\pi T(f_c + \Delta f)$ which will fall directly between each pair. Adding these two branches back together will give an approximation of the original spectrum back. Figure 8 follows these steps through for the triangle signal. Note that at the end we do not get a perfect triangle back because f_c was less than 4 kHz, so some of the high frequency is lost. We'll see that the speech will still be intelligible, if distorted, because of this lost high frequency content.

(e) First, I wrote a Matlab function as an `.m` file to implement the descrambler shown in Figure 7. The listing of this function is

```
function xhat = unscramble(y,omega1,omegac,domega,Nh)
% y is scrambled speech signal input
% omega1 is the DT frequency corresponding to f1 in the scrambler
% omegac is the DT frequency corresponding to fc in the scrambler
% domega is the DT frequency corresponding to delta f in the scrambler
% Nh is the order to use for the filters.
  hlpf = fir1(Nh,0.5);
  hhpf = fir1(Nh,0.5,'high');

  n = (0:(length(y)-1))';

  v1 = filter(hlpf,1,y);
  v2 = filter(hhpf,1,y);

  h2 = fir1(Nh,(omegac+domega)/pi);

  x1 = 2*cos((omegac+domega)*n).*v1;
  w1 = filter(h2,1,x1);

  x2 = ((-1).^n).*v2;
  w2 = filter(hlpf,1,x2);

  xhat = w1+w2;
```

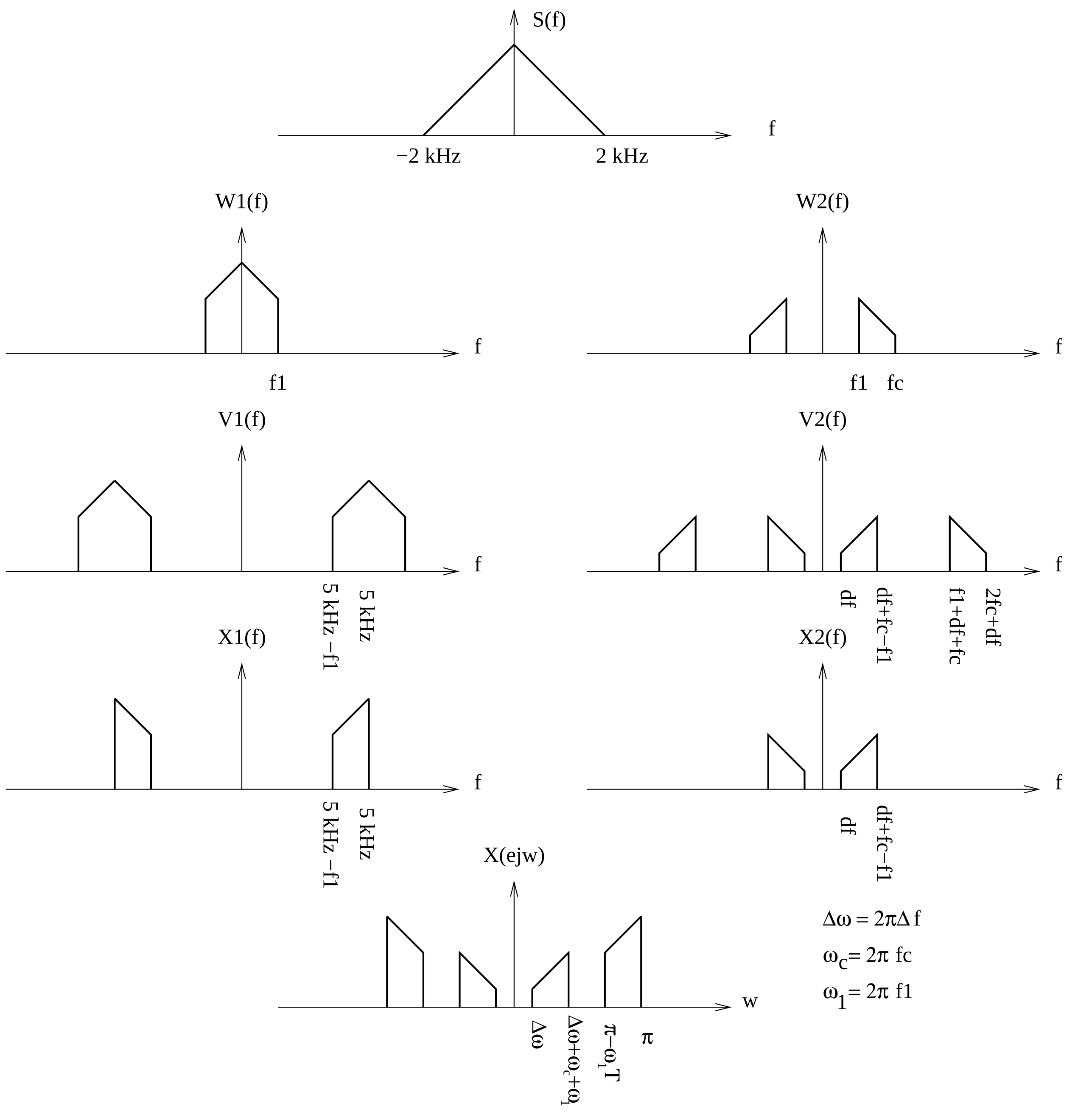

Figure 6: Test signal with triangle spectrum

With some trial and error, we can determine that $\Omega_1 = 0.14\pi$, $\Omega_c = 0.4\pi$ and $\Delta\Omega = 0.03\pi$ produces about as intelligible a signal as we can get with this system. The commands to descramble the signal are

```
[x,fs] = wavread('../assignment/group0.wav');
Nx = length(x);
n = (0:(Nx-1))';

omega1 = 0.14*pi;
omegac = 0.4*pi;
domega = 0.03*pi;
Nh = 100;
y = unscramble(x,omega1,omegac,domega,Nh);

y = y-mean(y);
y = y./max(abs(y));
wavwrite(y,fs,16,'matlab3test.wav');
```

The earth-shattering secret message is "Stuff those with soft feathers." At last, we understand the profound mysteries of the mind of the Renaissance genius Manicotti Ravioli.

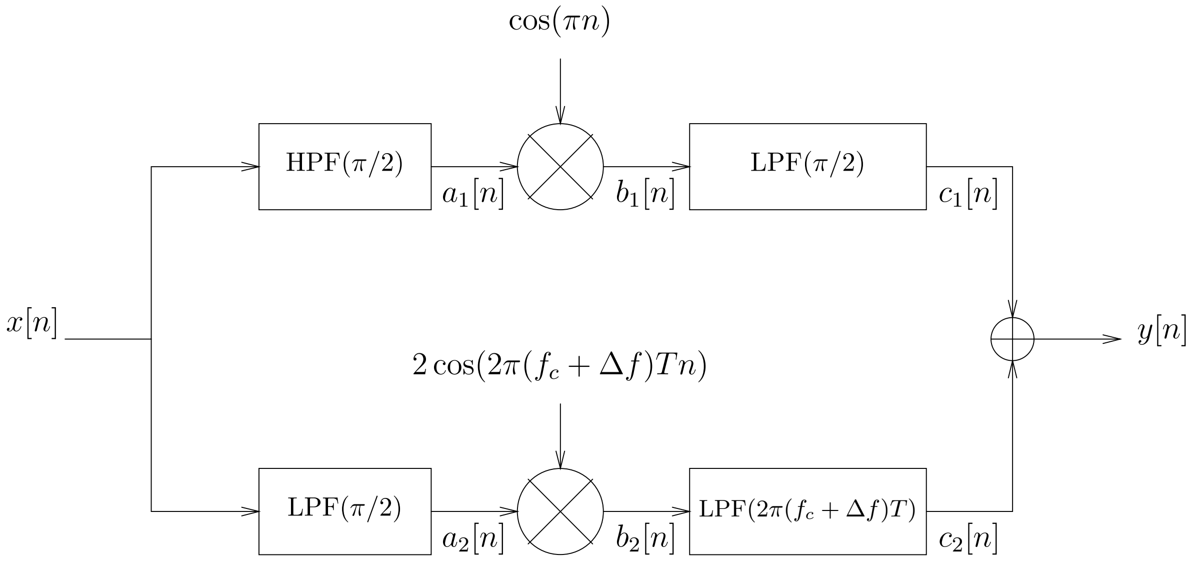

Figure 7: Proposed descrambling system

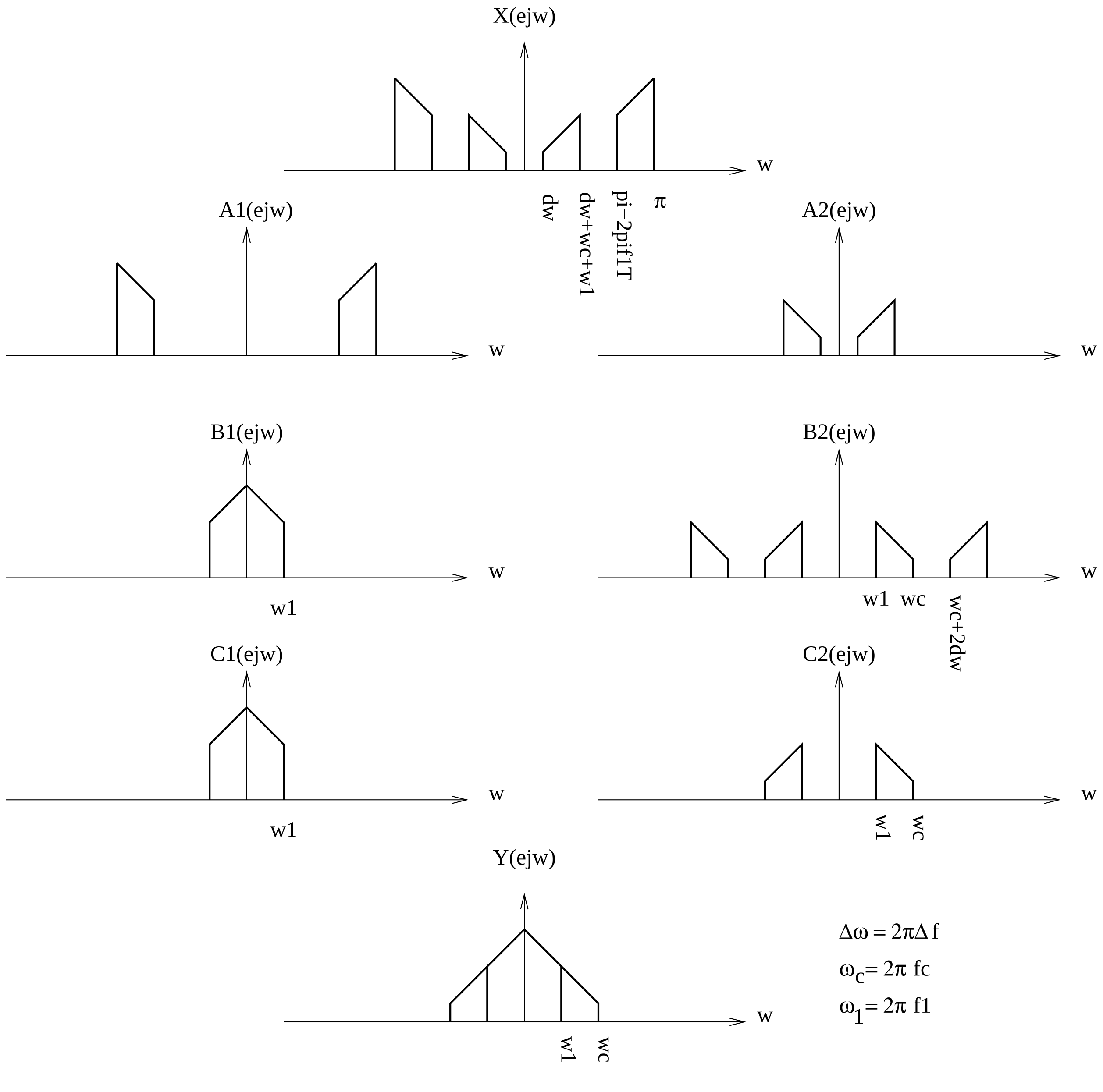

Figure 8: Descrambling triangle spectrum test signal