Alexander Hebert
ECE 6397
HW #2

1)

a) The file, signal.dat, was downloaded and loaded into the vector *x* in MATLAB.

b) For sampling frequency of $f_s = 80\ [Hz]$, the sampling period is $T_s = \frac{1}{80} = 0.0125\ [s]$.

A time vector, *t*, is created, corresponding to *x(t)* on the interval $0 \le t \le 4\ [s]$. A plot of *x(t) vs. t* is shown in Figure 1.

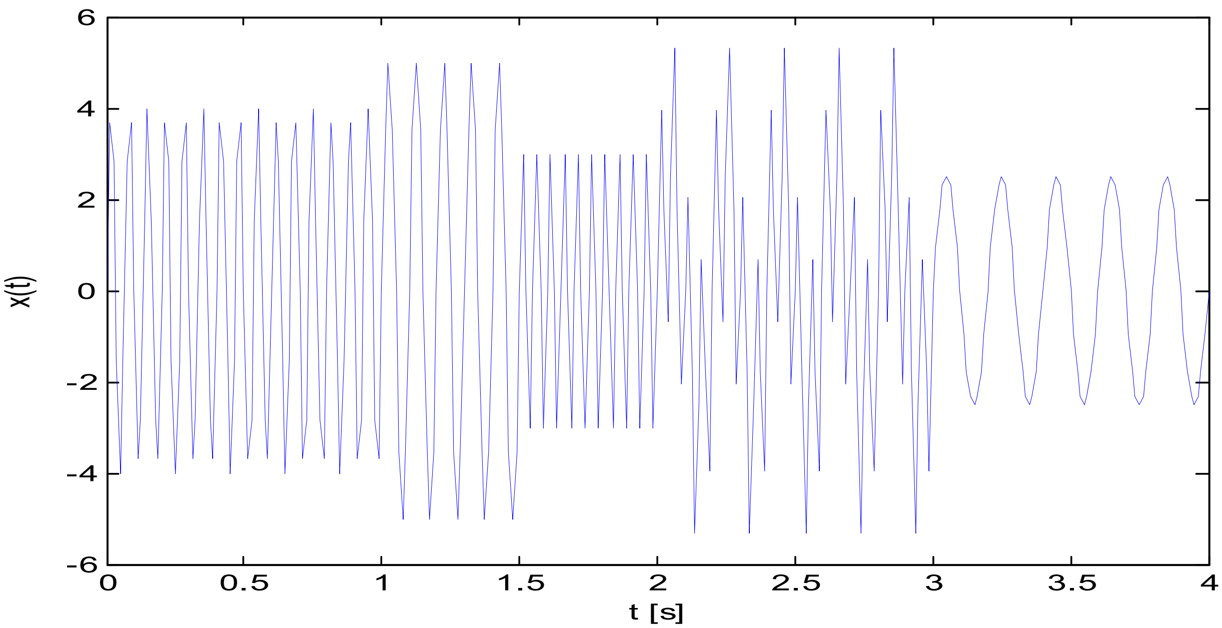

Figure 1. *x(t)* vs. *t* for f_s = 80 [Hz] on the interval $0 \le t \le 4$ [s].

c) In Figure 1, we observe a sequence of different frequencies on distinct intervals. For $0 \le t \le 1$ [s], the waveform appears to have 2-3 sinusoidal frequency components close in range to each other. For $1 \le t \le 1.5$ [s] and $1.5 \le t \le 2$ [s], the waveform has a single frequency component on each interval where the latter is higher frequency than the former. For $2 \le t \le 3$ [s], the waveform has a sum of at least two frequency components; one is low frequency, and the other is high. For $3 \le t \le 4$ [s], the waveform has a single frequency component, which is the lowest frequency of all the intervals.

2)

a) The FFT of *x(t)*, X, is computed in MATLAB using the *fft()* function. *fftshift*() is used to center the zero-frequency component.

b) The magnitude and phase of X are plotted vs. frequency in Figure 2.

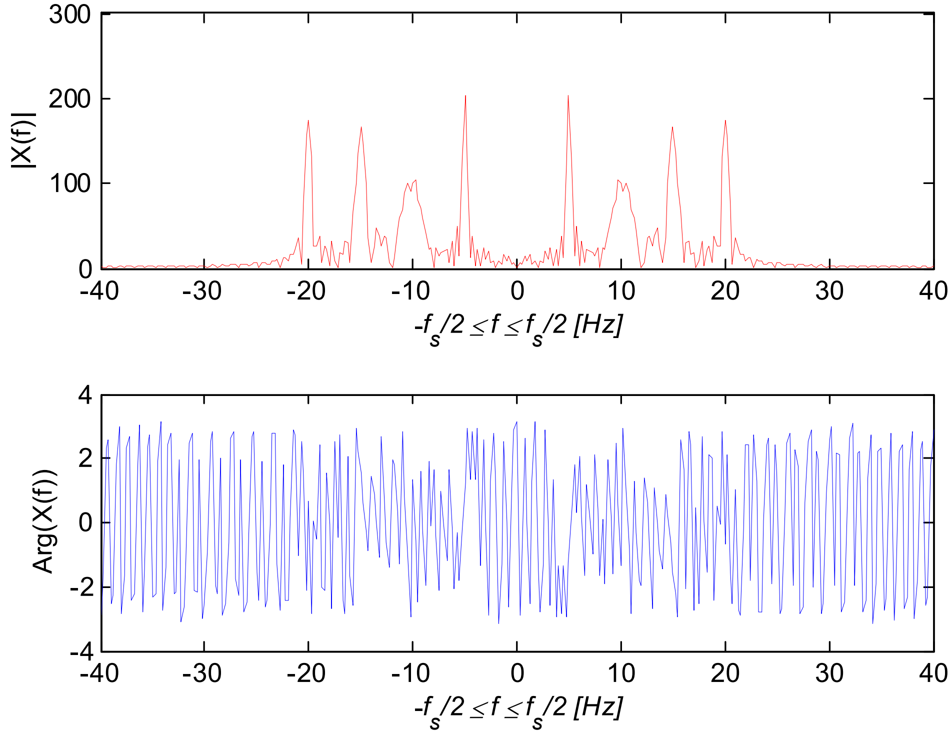

Figure 2. Magnitude and phase of X vs. frequency for $-f_s/2 \leq f \leq f_s/2$ [Hz] where $f_s = 80$ [Hz].

c) From the Fourier transform of X in Figure 2, the major frequency components are visible in the magnitude plot. The largest peak occurs at about 5 [Hz]. There are three frequency components close to each other (≈1 [Hz] apart) at roughly 8, 9, and 10 [Hz]. The other two peaks have similar magnitude and occur at 15 [Hz] and 20 [Hz]. Noise is present throughout the distribution but drops rapidly past 20 [Hz]. In the phase plot, there are transitions from 0 to 10 [Hz] and 10 to 20 [Hz]; their meaning is not understood.

3)

a) The short-time Fourier transform (STFT) is implemented as a MATLAB function in *stft.m*.

b) The output matrix, *S*, from *stft.m* is input into *spectrogram.m* to compute and plot the spectrogram of *x(t)* for five values of Gaussian-window spread σ (see Figures 3 – 7).

For σ = 0.025:

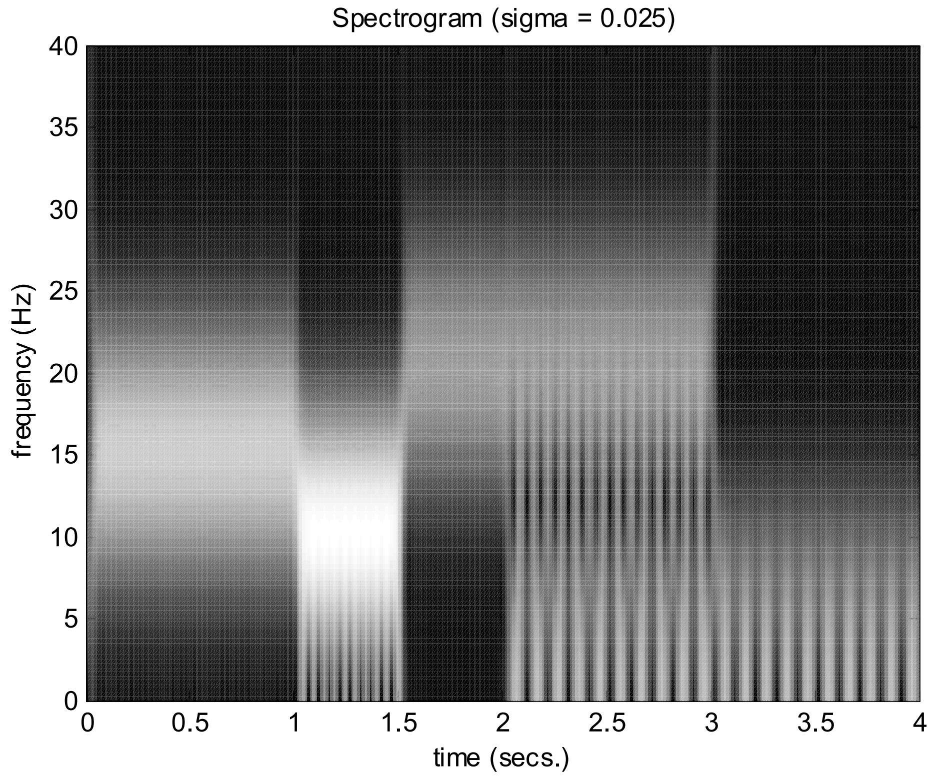

Figure 3. Spectrogram of *x(t)* for σ = 0.025 and f_s = 80 [Hz].

For σ = 0.05:

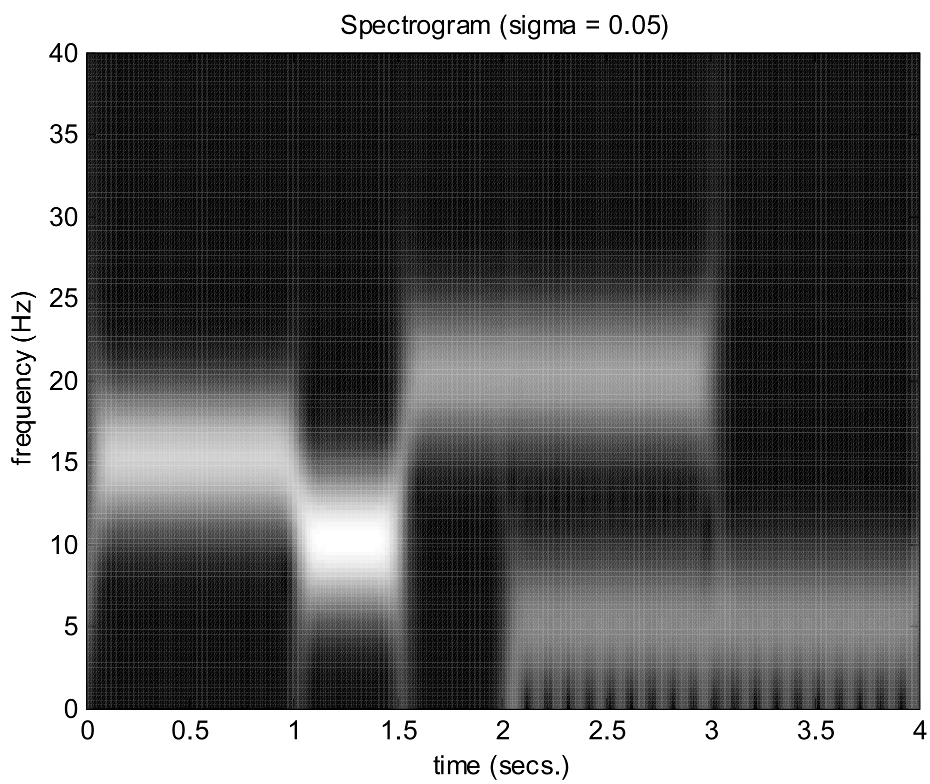

Figure 4. Spectrogram of *x(t)* for σ = 0.05 and f_s = 80 [Hz].

For σ = 0.1:

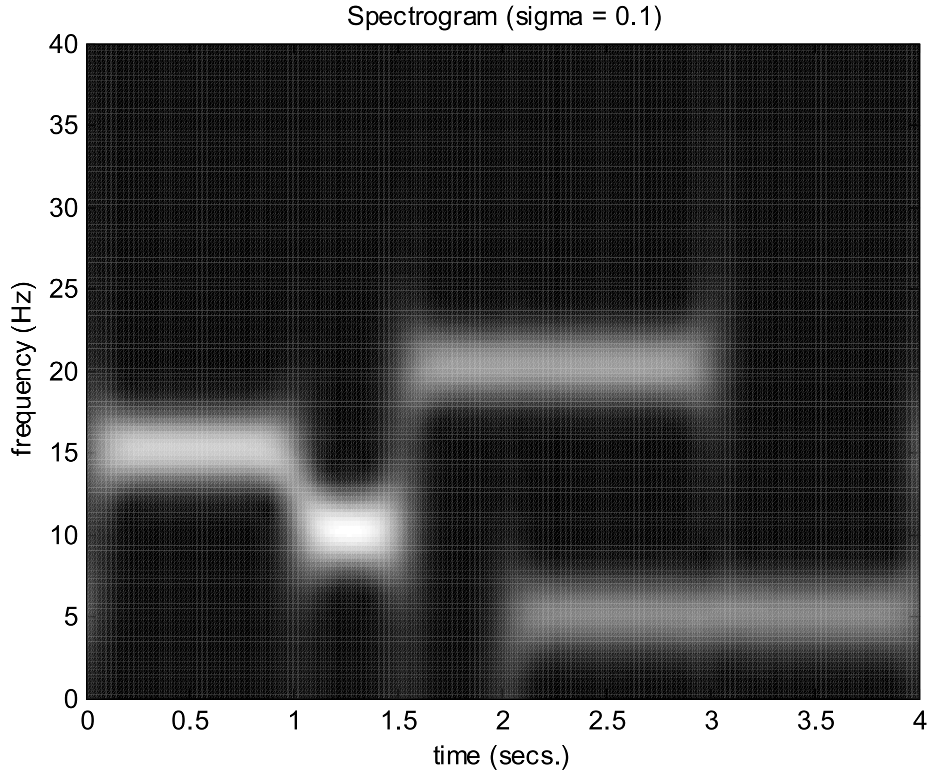

Figure 5. Spectrogram of *x(t)* for σ = 0.1 and f_s = 80 [Hz].

For σ = 0.2:

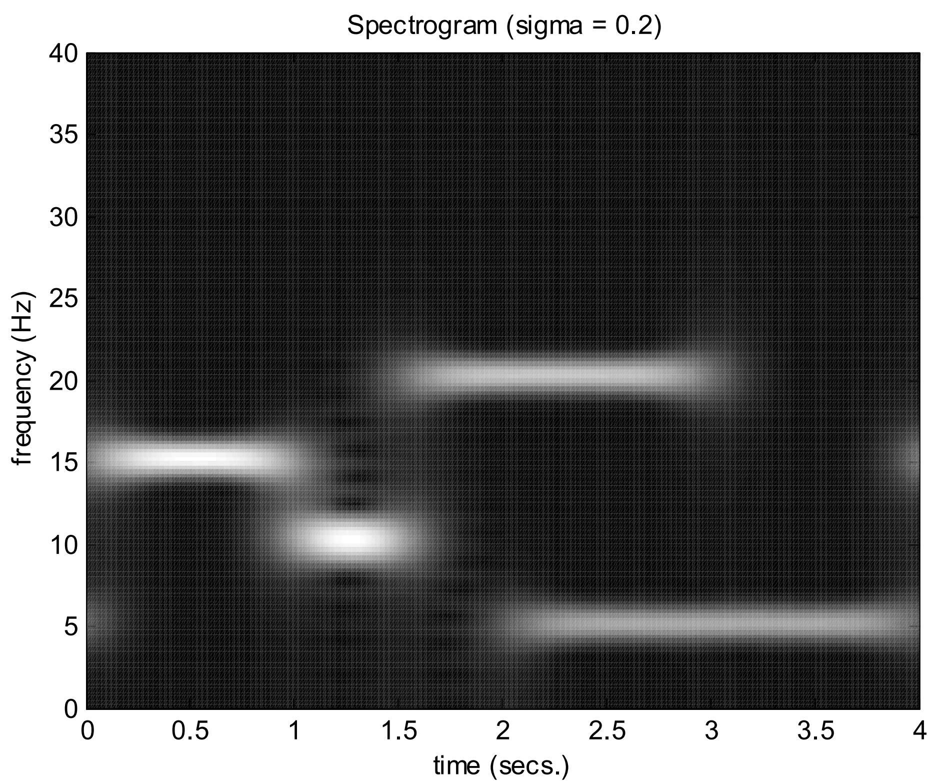

Figure 6. Spectrogram of *x(t)* for σ = 0.2 and f_s = 80 [Hz].

For σ = 0.5:

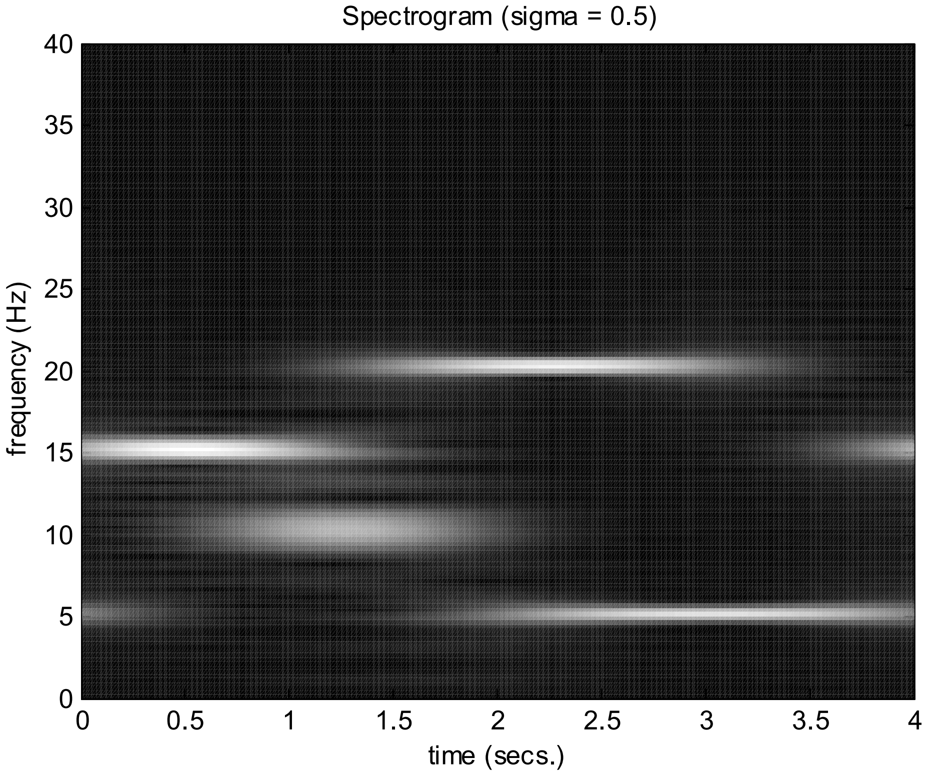

Figure 7. Spectrogram of *x(t)* for σ = 0.5 and f_s = 80 [Hz].

c) From Figures 3 – 7, we observe that time and frequency resolution depend on σ. Small σ corresponds to a narrow window. In Figure 3 with σ = 0.025, time resolution is very good, showing the time intervals clearly; consequently, frequency resolution is poor. Large σ corresponds to a wide window. In Figure 7 with σ = 0.5, frequency resolution is very good, showing the frequency components clearly; consequently, time resolution is poor. In Figure 5 with σ = 0.1, time and frequency resolution are both moderate, resulting in an in-between trade-off. Heisenberg's Uncertainty Principle and the law of time-frequency trade-off are evident. The choice of σ would depend on where resolution is needed.

Acknowledgements

The MATLAB files *makewindowmatrix.m* and *spectrogram.m* were used to complete this assignment.

Appendix

The following MATLAB code was written for this assignment.

stft.m

```
function S = stft(x,fs,sigma)
% STFT: short-time Fourier transform
% parameters:
% x (column vector) is the time domain signal with length N
% fs is the sampling frequency
% sigma is the spread for the Gaussian window
% S is an NxN matrix with N^2 samples of the time-frequency plane.
%  The vertical dimension is for frequency, and the horizontal
%  dimension is for time.

% Gaussian window
N = length(x);
Ts = 1./fs;
T = Ts*(N-1);
t = [-0.5*T:Ts:0.5*T]'; % -T/2 <= t <= T/2
g = (1./((pi*sigma^2)^(1/4))).*exp((-1*t.^2)./(2*sigma^2));

% G is an NxN matrix with all N possible cyclical time shifts of g.
G = makewindowmatrix(g);

% X is an NxN matrix with signal x windowed by G for corresponding columns
X = zeros(N,N);
for k = 1:N
   X(:,k) = x.*G(:,k);
end

% S is an NxN matrix of the FFT of X
S = fft(X);
```

hw2.m

```
% Alexander Hebert
% ECE 6397
% HW #2
clc; clear; close all;

% 1a
x = (load('signal.dat', '-ascii'))';
N = length(x);

% 1b
fs = 80;
Ts = 1./fs; % period for sampling freq f = 80 Hz
T = Ts*(N-1);
t = [0:Ts:T]';
figure(1)
plot(t,x)
xlabel('t [s]');
ylabel('x(t)');
```

```matlab
% 2a
X = fftshift(fft(x));

% 2b
Xmag = abs(X);
Xphase = angle(X);
f = linspace(-fs./2,fs./2,N);
figure(2)
subplot(2,1,1);
plot(f,Xmag,'r');
xlabel('-\itf_{\its}/2 \leq \itf \leq \itf_{\its}/2 [Hz]');
ylabel('|X(f)|')
subplot(2,1,2);
plot(f,Xphase,'b');
xlabel('-\itf_{\its}/2 \leq \itf \leq \itf_{\its}/2 [Hz]');
ylabel('Arg(X(f))');

% 3b
sigma = [0.025,0.05,0.1,0.2,0.5];

for k = 3:7
    S = stft(x,fs,sigma(k-2));
    figure(k)
    spectrogram(S,fs,sigma(k-2));
end
```