Stanford University
EE 264: Digital Signal Processing
Summer, 2018

Homework #06

Date assigned: August 6, 2018 Date due: August 15, 2018

Reading: This assignment covers primarily lectures 10 to 12, which correspond to chapters 8, and 10 of the textbook **DTSP 3e**.
Homework submission: Please submit your solutions on Gradescope. Create a single .pdf file containing all your analytical derivations, sketches, plots, and Matlab code (if applicable).
Extensions and late submissions: To ensure that we can release the solutions of this assignment in a timely manner for the final exam, we cannot grant extensions or accept late submissions.

Problem 1: Adapted from the final exam of EE 373A – Winter 2018 (40 points)

General description

As discussed in class, we can use adaptive filters to cancel noise or interference from a desired signal by appropriately filtering a correlated noise measurement. In this exercise, you will build a typical noise canceling system to minimize the noise from an audio signal. Figure 1 shows the diagram of noise generation. The speaker produces a signal $s[n]$, like a song, for instance. Only the front microphone captures the signal produced by the speaker. But both microphones capture the noise generated by a noise source. The transfer function $H(z)$ is unknown, and it relates the noise in the rear microphone to the noise in the front microphone. To keep things simple, we will assume that this transfer function is linear and time invariant.

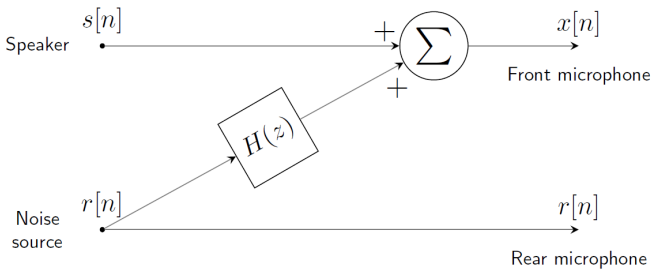

Figure 1: Model for noise generation.

The adaptive noise canceler is shown in Figure 2. The noise from the rear microphone is filtered by the adaptive filter and added to the signal captured by the front microphone. Ideally, the signal $y[n]$ would perfectly cancel the noise component of $x[n]$.

You're given the audio signal $s[n]$ in the file `guitartune.wav`. $H(z)$ is given in the file `problem1_template.m`. The noise $r[n]$ is a white, zero-mean Gaussian noise with variance $\sigma_r^2 = 2 \times 10^{-3}$. This value of variance is chosen so that the signal-to-noise ratio (SNR) at the

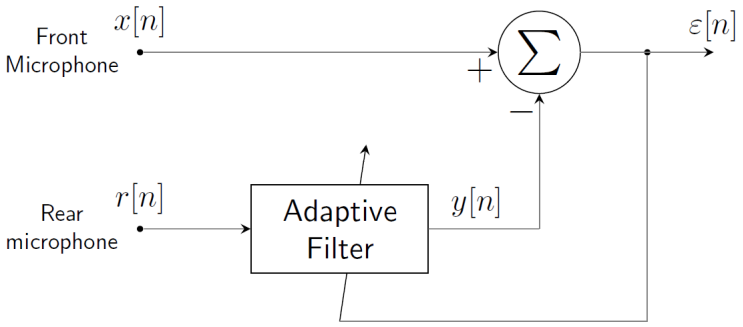

Figure 2: Typical noise canceling system.

front microphone is approximately 10 in linear units (or 10 dB). Assume that the adaptive filter has a total 50 weights. To generate the corrupted signal at the front microphone $x[n]$ follow the diagram of Figure 1.

Based on the given information, please answer the following questions. You may use the Matlab file `problem1_template.m`. as a template to develop your answers. This file also contains some useful Matlab hints. Please include your code in your solutions file.

(a) Give an expression for the transfer function of the adaptive filter that would result in perfect noise cancellation. Your answer should depend on $H(z)$ only.

(b) What is the Wiener solution for the weights of the adaptive filter if $x[n]$ and $r[n]$ were uncorrelated? In other words, what would be the resulting adaptive filter if there were no noise component in $x[n]$?

(c) Calculate the adaptation constant μ such that $\mu = 0.001\mu_{max}$, where $\mu_{max} = 1/\text{trace}(R)$.

(d) Using the value of μ you computed in part (c), estimate how long it takes, in seconds, for the LMS algorithm to converge. You may assume that convergence is achieved after 4 time constants. The sampling frequency of the audio signal is 22.05 kHz.

(d) Describe whether the following metrics would increase, decrease, or remain the same, if we increase the value of μ. Please justify your answers with one or two sentences.

- Convergence time (learning curve time constants)
- Minimum mean square error (ξ_{min})
- Excess mean square error

(e) Implement the adaptive noise canceling system shown in Figure 2. In addition to your code, turn in the following plots:

- The mean square noise $(s[n] - \varepsilon[n])^2$ averaged over 20 independent runs. The x-axis of your plot should be time in seconds.
- On a separate graph, plot the coefficients of the converged adaptive filter you obtained at the last iteration of the last run.

- On another graph, plot the magnitude and phase response of the converged adaptive filter. Compare it to the magnitude and phase response of the ideal noise canceler you proposed in part (a). Explain any discrepancies.

Use the Matlab function `sound` to play the input signal to the noise canceler $x[n]$, and the noise-canceled signal $\varepsilon[n]$. For the latter, you should be able to hear the noise progressively becoming smaller.

Matlab hint: the Matlab functions `impz` and `freqz` are useful for plotting impulse response, and magnitude and phase responses of discrete-time systems.

Problem 2: DFTs of simple sequences (20 points)

Given these 8-point signals $x[n], n = 0, \ldots, 7$, compute the 8-point DFTs of $X[k], n = 0, \ldots, 7$. Only minimal computation should be required. For example, once you find the answer for (a), you can use DFT properties to find the answers for (b), (c) and (d). Similarly, once you find the answer for (e), you can easily find the answer for (f).

(a) $x[n] = \{1, 1, 1, 1, 1, 1, 1, 1\}$

(b) $x[n] = \{1, -1, 1, -1, 1, -1, 1, -1\}$

(c) $x[n] = e^{jn\pi/2}, n = 0, \ldots, 7$

(d) $x[n] = \sin(\pi n/2), n = 0, \ldots, 7$

(e) $x[n] = \{1, 0, 0, 0, 0, 0, 0, 0\}$

(f) $x[n] = \{0, 0, 1, 0, 0, 0, 0, 0\}$

Problem 3: DFTs (15 points)

Consider the six real signals $x_1[n], \ldots, x_6[n]$ shown in Figure 3, and the six DFTs $X_a[k], \ldots, X_f[k]$ shown in Figure 4, each of length $N = 8$. Match each signal to its DFT. Provide a table like the following, filling the appropriate letters in the second column. Provide a brief justification based on symmetry, slope of the phase $\angle X[n]$, d.c. value $X[0]$, etc.
Hint: aside from looking at the table of DFT properties, you may find it helpful to think about the periodic extension $\tilde{x}[n]$ of each $x[n]$.

Signal	DFT	Justification
1		
2		
3		
4		
5		
6		

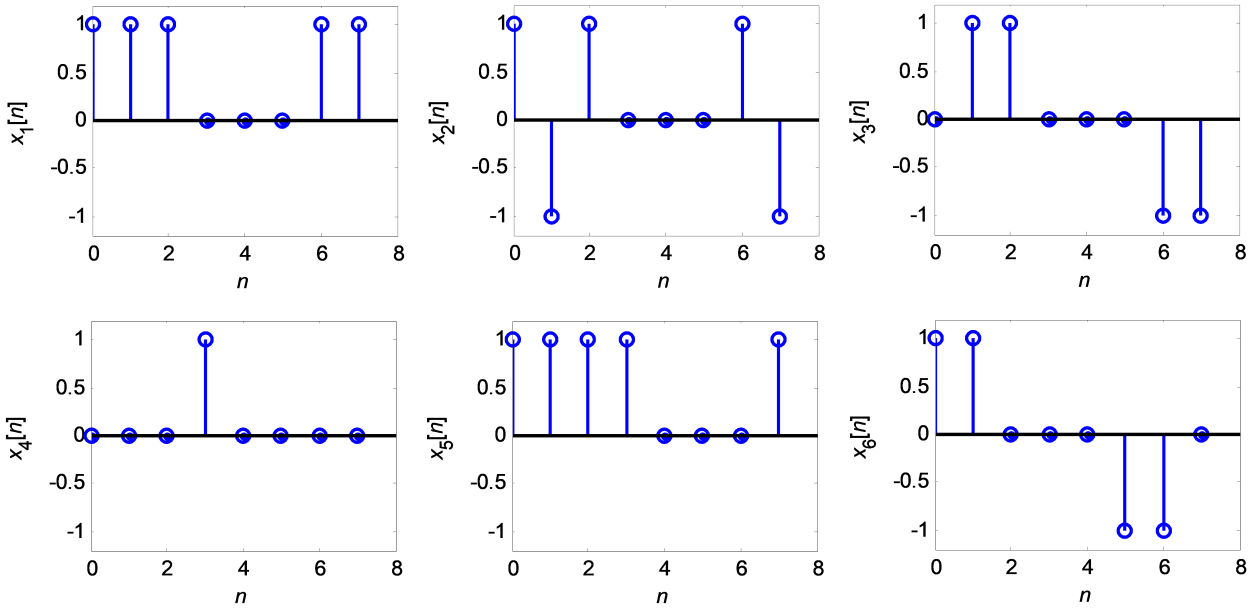

Figure 3: 8-point real signals $x_1[n], \ldots, x_6[n]$

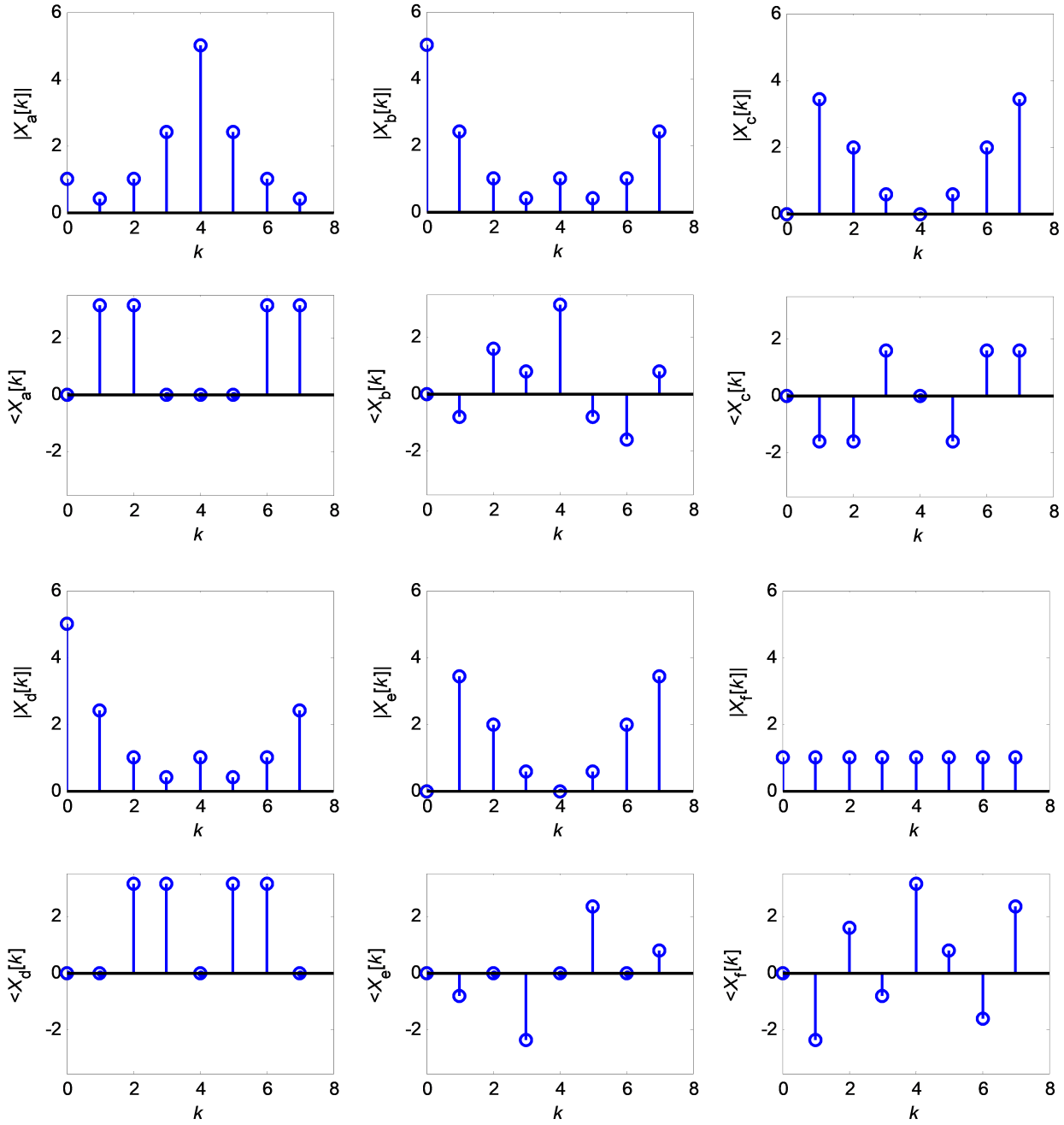

Figure 4: Magnitude ($|\cdot|$) and phase ($\angle\cdot$) of DFTs $X_a[k], \ldots, X_f[k]$

Problem 4: Linear convolution and circular convolution (20 points)

(a) Given a L-point sequence $x[n]$ and a P-point sequence $h[n]$, under what conditions would the N-point circular convolution match the linear convolution i.e, $x[n] * h[n] = x[n] \textcircled{N} h[n]$?

(b) Several times during the course we used the Matlab function `xcorr` to estimate the autocorrelation or the cross-correlation function of signals. Without any normalization, `xcorr` computes the the deterministic cross-correlation function, which can be expressed as a linear convolution:

$$c_{xy}[m] = \sum_{l=-\infty}^{\infty} x[m+l]y^*[l] = x[m] * y^*[-m] \tag{1}$$

Assuming that both $x[m]$ and $y[m]$ are L-point sequences ($x[m] = 0, m < 0$ or $m \geq L$ and $y[m] = 0, m < 0$ or $m \geq L$), propose a method of calculating $c_{xy}[m]$ using FFTs and IFFTs. Give an equation for $c_{xy}[m]$ using your method and specify the length N of the FFTs/IFFTs involved.

Hint: Don't forget to use the conjugate and time reversal properties of the DFT.

Note: If you want to test your method numerically, remember that according to the FFT/IFFT convention, your answer will be indexed from 0 up to $N-1$. However, $c_{xy}[m]$ is defined from $-(N-1)/2$ up to $(N-1)/2$. Hence, you should use the command `fftshift` to match your method with the output of `xcorr`. Answers with and without `fftshift` will be accepted.

(c) How would your method be simplified if instead of the cross-correlation function, we were calculating the autocorrelation function i.e., $y[n] = x[n]$?

Problem 5: Overlap-add or overlap-save? (25 points + 10 points extra)

(a) Use either the overlap and add or overlap and save method to implement the notch FIR filter you designed in Homework #5 to filter out the 60 Hz interference of the ECG signal. On the same graph plot the outcome of your implementation, the result of filtering with the function `filter`, and the "clean" ECG signal. Specify the block length L and the FFT size N that you chose in your implementation. You may want to read parts (b) and (c) before choosing these values.

Note: It is not important for this question whether you designed a good notch filter or not. If you prefer, you can also use the FIR filter provided in the solutions of Homework #5. However, the main point of this question is to understand the implementation of block convolution through overlap and add or overlap and save methods.

(b) Compare the number of multiplications of your method with a direct implementation of an FIR filter. For this comparison, use the following metric

$$C = \frac{\#\text{ of multiplications}}{\#\text{ of useful output samples}}. \tag{2}$$

According to this metric, a direct implementation of an FIR filter of order $M = 52$ has complexity $C = 53$, since it requires 53 multiplications per each useful output sample. On the other hand, each FFT requires $\approx 2N(\log_2 N - 1)$ multiplications, but remember that not all N outputs are useful in the overlap and add or overlap and save methods.

Note: Don't be surprise if the number of operations required by block convolution isn't much smaller than the direct FIR filter. This filter is not long enough for the FFT to yield significant savings in computation.

(c) Give an expression for C in terms of the filter order M and the FFT length N. This equation should not depend on whether you chose overlap and add or overlap and save. On a single graph, plot C as a function of N when $M = 50$, $M = 100$, and $M = 150$. To facilitate visualization, for each value of M, plot C using N in a range from $2M$ up to 5000. For simplicity, your plot may include non-integer values of N.

Note: This exercise should show you that for each filter order M there is an optimal FFT size N. The optimal block size L then follows from knowing N and M.

(d) Optional (extra 10 points) Implement the other method that you did not implement in part (a).

Problem 6: Spectrograms (Question 10.32 of the textbook) (15 points)

An analog signal consisting of a sum of sinusoids was sampled with sampling rate of $f_s = 10000$ samples/s to obtain $x[n] = x_c(nT)$. Four spectrograms showing the time-dependent Fourier transform $|X[n, \lambda)|$ were computed using either a rectangular window or Hamming window. They are plotted in the figure below.

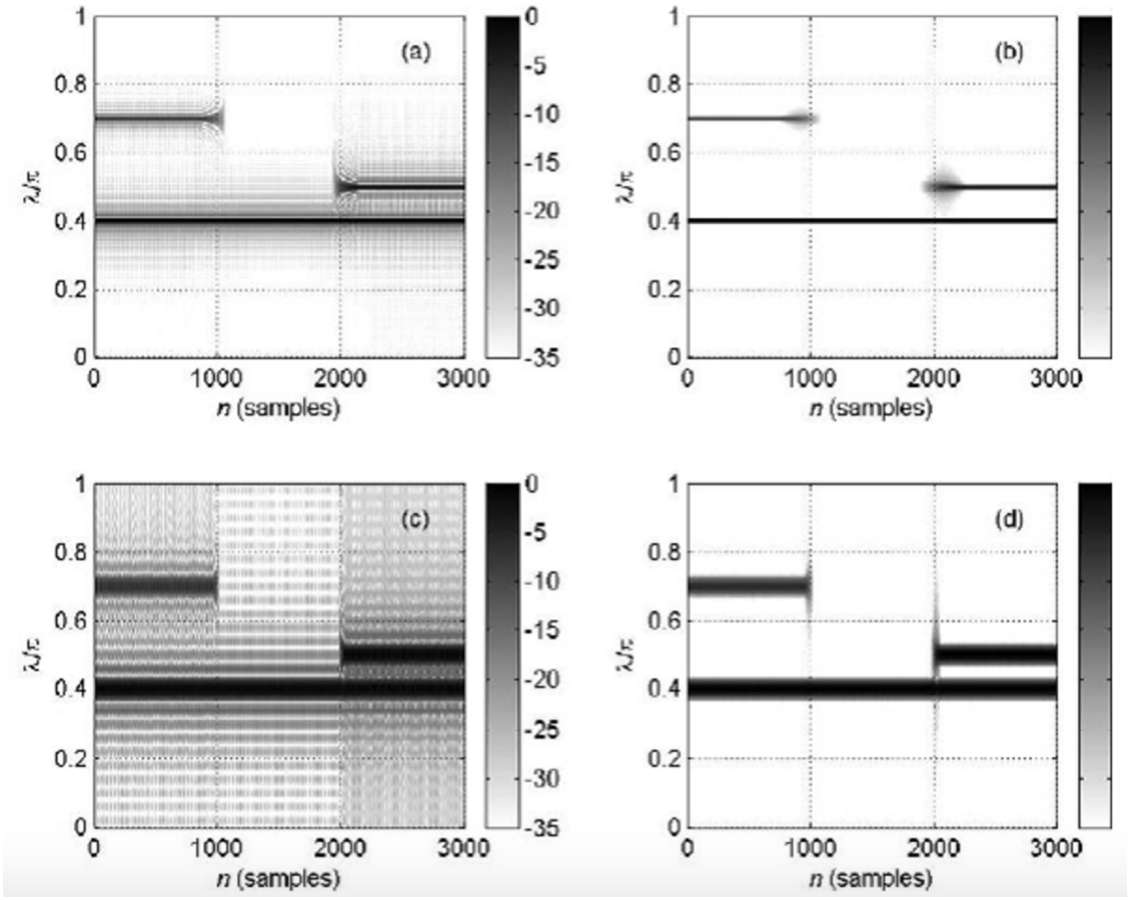

Figure 5: Spectrograms of $|X[n, \lambda)|$. The y-axis of all spectrograms is λ/π. The magnitude of the spectrograms is plotted in log scale and only the top 35 dB is shown.

Answer the following questions and give a brief justification to support your answers.

(a) Which spectrograms were computed with a rectangular window?

(a) (b) (c) (d)

(b) Which pair or pairs of spectrograms have approximately the same frequency resolution?

(a&b) (b&d) (c&d) (a&d) (b&c)

(c) Which spectrogram has the shortest time window?

(a) (b) (c) (d)

(d) Write an equation for the continuous-time signal $x_c(t)$ that when sampled at sampling rate of $f_s = 10000$ samples/s would produce the above spectrograms. Be complete as you can in your description of the signal and indicate any parameters that cannot be obtained from the spectrograms.

Problem 7: Frequency modulation (FM) (extra 20 points)

A frequency-modulated signal with sinusoidal modulation is defined by the equation

$$x_{FM}(t) = \cos(\theta(t)) = \cos(\Omega_c t + \beta \sin(\Omega_m t)), \tag{3}$$

where Ω_c is the carrier frequency, Ω_m is the modulating frequency, and β is called the modulation index. In FM transmission, as in radios, the modulating frequency Ω_m is changed according to the signal being transmitted e.g., a song.

(a) Write an equation for the instantaneous (radian) frequency defined by

$$\Omega_i(t) = \frac{d\theta(t)}{dt}. \tag{4}$$

Why is the quantity $\beta\Omega_m$ called the maximum frequency deviation? For frequencies $\Omega_c = 2\pi(1000)$ and $\Omega_m = 2\pi$, either sketch or plot $\Omega_i(t)/(2\pi)$ for $0 \leq t \leq 4$ seconds, for $\beta = 0.2$ and for $\beta = 500$.

(b) Show that if $\beta << \pi/2$, we can approximate $x_{FM}(t)$ as

$$x_{FM}(t) \approx \cos(\Omega_c t) - \beta \sin(\Omega_m t) \sin(\Omega_c t) \tag{5}$$

This is called the *narrowband FM approximation*. Either write an equation or a description of the Fourier transform $X_{FM}(j\Omega) = \mathcal{F}\{x_{FM}(t)\}$. Your description could be something like "there will be impulses at frequencies...".

(c) The FM signal in (3) with $\Omega_c = 2\pi(1000)$, $\Omega = 2\pi$ is sampled with a sampling rate of $F_s = 1/T = 8000$ samples/s. Give an equation, leaving β as a parameter for the samples $x[n] = x_{FM}(nT)$.

(d) In Matlab, generate 32768 samples of the FM signal in part (c) for the case $\beta = 0.2$. Listen to the signal using the function `soundsc()`.

(i) Compute a single 32768-point DFT of the signal (use a rectangular window). Make a plot of the DFT magnitude. Label the frequency axis in terms of analog frequencies: $-Fs/2 \leq F \leq Fs/2$ in Hz.

Note: It will be helpful to use the command `axis([990 1010 0 0.5])` to look only at the frequencies in the band $990 \leq F \leq 1010$ Hz. Is what you see consistent with your answer to part (b)?

(ii) Use the Matlab function `spectrogram` to plot the spectrogram of the sampled FM signal. Use a Hamming window of length 256 with overlap of 250 samples. Does your spectrogram show the variation of instantaneous frequency that you plotted in part (a)? If not, why not?

(e) Now generate 32768 samples of the FM signal in part (c) for the case $\beta = 500$. Listen to the signal using the function `soundsc()`.

(i) Compute a single 32768-point DFT of the signal (use a rectangular window). Make a plot of the DFT magnitude. Label the frequency axis in terms of analog frequencies: $-Fs/2 \leq F \leq Fs/2$ in Hz.

(ii) Use the Matlab function `spectrogram` to plot the spectrogram of the sampled FM signal. Use a Hamming window of length 256 with overlap of 250 samples. Does your spectrogram show the variation of instantaneous frequency that you plotted in part (a)?

(iii) Experiment with window length and overlap to see how these parameters affect the image. Specifically, take a look at windows that are much shorter and also much longer than 256. How would the choice of window length depend upon the frequencies Ω_c and Ω_m if we wish to track the time variation of the instantaneous frequency with good resolution?