BMEn 8101: Homework Set #8 **Instructor: Professor Hubert Lim**

Due May 7, 2024 by 11:59PM via email to the gmail address (BMEN8101Lim@gmail.com). This assignment will be worth double homework points so 60 total points.

For this assignment, you will download the 3 .mat files from the moodle site: Sig.mat, SigNoisy.mat, SigLessNoisy.mat. Each will have a signal vector as well as the sampling frequency, fs.

Your goal is to denoise the noisy signal to get it as close as possible to the original signal (SigOG). One of the noisy signals has a much lower SNR than the other one so you can compare performance for different SNR levels.

I have also uploaded Matlab code called 'WienerScalart96.m', which was written by Esfandiar Zavarehei and downloaded from an online website. You can immediately use this function by inputting at the Matlab prompt:

```
output=WienerScalart96(signal,fs,IS)
```

where 'signal' would just be your signal vector, and IS corresponds to the time of the noise-only portion at the beginning of the signal, which is ~0.15 seconds for your signals. You can even set 'signal=SigOG' to see if the algorithm actually modifies your signal when just inputting the original signal.

This code includes several sub-functions to run 'WienerScalart96' and please go through all the code and understand each line and how the algorithm is implemented to denoise the signal. You will learn quite a bit from understanding the code and how to actually implement all stages, not only of the Wiener filter, but in performing short-time Fourier Transform and windowing/reconstruction. This code incorporates many of the concepts covered over the semester. I have included some comments to help you understand the code and where you can make changes to the code. **You MUST modify the code to perform the list of items below and please use this code only to create your own custom program (i.e., you can't just copy/implement this provided code)**. Keep in mind that this code performs all the components of the flow chart that was drawn in class. The main difference in this code and what we derived in class is that the method for estimating the frequency-domain version of the Wiener filter in the code is done by a recursive method (Decision-Directed method) instead of the spectral subtraction method. You will need to implement and compare both methods. Basically, you will comment out the Decision-Directed code and input your own code to do the spectral subtraction method. Keep in mind that the Decision-Directed Method is just a way to take past SNR(k) values and combine it with the newest estimate for the SNR(k) value and sum them together with some weighting. This weighting allows you to decide if you want to put more weight on past or more on present values of SNR(k). If you put more weight on past values, then it results in a more smoothed version of the SNR(k) over time. Please read over the extra notes on this topic provided in the CANVAS/HW#8 files.

Please investigate the items below and provide plots and explanations to justify your conclusions. Keep in mind that the items below overlap with each other so you would try a combination of them together to come to some conclusions of how the Wiener filter performs and what parameters seem to work well for the signals. You will need to save the denoised signal as wav files as you did before and then can listen to them in order to compare performance across parameters. You will need to email the T.A. at the address: **BMEN8101Lim@gmail.com** with your best denoised .wav file with this format: **HW8_StudentName_SigNoisy.wav** and **HW8_StudentName_SigLessNoisy.wav**. **You also need to email your written assignment and Matlab code supporting all your analyses and interpretations**

for this assignment in that email. You can include scanned documents in your submission. Please show all of your work to justify your answers and that you understood all of the requests below.

I also have included my final version of the code (`WienerHver2.m`) for doing this HW#8. Please only look over this if you can't figure things out on your own (**AS A LAST RESORT**) and afterwards to check your code so that you understand how to do the assignment. I figured this would help you understand the algorithms/coding more efficiently and to make sure you are doing things correctly. Please do not copy my code exactly when submitting your Matlab code. You need to provide your own Matlab code so we can check that you did in fact develop your own code and tried different options/parameters! Please make sure to look at how I implemented the Decision-Directed Method in Matlab in `WienerHver2.m` by comparing with the theoretical notes from class.

Things to do:

1) Compare the Decision-Directed Method vs the Spectral Subtraction Method.
Make sure to try using ONLY the initial segment before the speech starts to estimate the Noise PSD. You will need to modify the code to do this since the current code updates the Noise PSD whenever it thinks there is a noise-only segment across frames. Actually, updating the Noise PSD in this way worsens the denoised signal than if you just use the initial segment. This is because the added noise is stationary and doesn't change over time (and is simple white noise). When the algorithm tries to update the Noise PSD over time, it actually takes many segments that are really soft speech segments that distorts the Noise PSD rather than improves it (also it can be difficult to predict which segments are only speech or only noise). Thus you need to code and show how just using the initial segment for Noise PSD estimation does a better job. If the noise was actually changing over time, then using this updating method in the current code would do a better job.

2) The code uses a Hamming window. You can try different windows. They use 40% overlap which doesn't necessarily follow the overlap constraint criterion covered in class. Download and look over the document 'Overlap Window Constraint' to see how to ensure the correct overlap is used. Use a small overlap and a large overlap to see if this helps at all. Also try a small window length (much smaller than the stationary components of the speech signal which you can ~determine by zooming in on the signal segments) and a larger window (much larger than the stationary effective length). You can play around with different lengths to see how this affects your denoised signal.

3) Try using different methods for estimating your PSD for your noise-only signal as well as your noisy signal to then calculate your Wiener filter based on equation (5) from class. In other words, you can just use the instantaneous values (i.e., FFT( )^2) at each time segment (i.e., frame) or you can use some averaged versions across time frames. The code uses the Welch Method PSD for calculating the PSD for the noise-only component because it is taking overlapping-windowed versions and averaging them together over time. The code uses a smoothed version across segments for the noisy signal PSD to then get the cleaned-signal PSD estimate based on the Decision-Directed Method. You will need to implement your own code for the spectral subtraction method and try smoothing out your noisy signal PSD across segments or even by cutting up a given segment into smaller segments to obtain an averaged PSD. Remember that you are facing a trade-off between greater variance but more frequency resolution for the periodogram versus less variance but less frequency resolution (more bias) for the averaging versions (such as Welch method or averaging over time frames as done already in the code).

4) Compare the results for the different SNR signals to see how SNR affects performance. In theory, the Wiener filter should perform substantially better as the SNR decreases. What do you notice about the quality of the filtered output based on both visual inspection and audio assessment of the denoised

signals? Even though the signal appears to be better denoised visually, our hearing system can still pick up minor errors in the signal so it is very difficult to fully restore a clean version of the original signal. For "fun", even try to apply your denoising algorithm on your original non-noisy signal and see how it will still cause some distortion in your signal since the algorithms are not perfect.

Keep in mind that the noisy signals contain white noise with constant power across frequencies. However, if there was colored noise (so different power values for different frequency bands), we could implement a more sophisticated method to use different weighted versions of the Wiener filter for different frequency bands. Also note that the spectral subtraction method is a very crude method and there are many more sophisticated methods based on modeling of the PSDs, maximum likelihood estimators for the wiener coefficients using the PDFs of the signals (as we did with the Decision-Directed SNR method), etc. But the spectral subtraction method combined with the Wiener filter still does a decent job for many signals and is the foundation for other more complex/hybrid methods that you can further investigate for your own research.

Even for the high SNR noisy signal, you should be able to do a decent job at denoising it with the Wiener filter since I only added stationary white noise to the signal. The Wiener filter is actually well-designed for handling additive white noise that is uncorrelated to the clean/desired signal. Visually you will see it does a pretty good job but when you actually listen to it, you can still hear some distortions even for the best parameters (i.e., because our ears pick up even the slightest distortions even though you can't see it visually in the signal). For the less noisy signal, you should be able to do a very good job at denoising it, especially with the Decision-Directed (SNR) method.