ECE446: Sensory Communication
Digital Signal Processing Problem Set Report

Pranshu Malik
1004138916

1 Manipulation of Signals in the Frequency Domain

To construct the signal in the frequency domain, we need to know the Fourier transform of the signal under consideration. The signal we want to construct are phasors, which are fundamentally cosine signals, i.e. $\text{Re}(Ae^{jf(t,\phi)}) = A\cos(f(t,\phi)) = Ae^{jg(\phi)} = \tilde{A}$ for some functions f and g. The continuous-time continuous-frequency Fourier transform (CTFT) for such a cosine is given by $\mathcal{F}\{\cos(\Omega_0 t)\} = \pi(\delta(\Omega - \Omega_0) + \delta(\Omega + \Omega_0))$. Now, to be able to use this transform on a computer, we have to use the discrete-time discrete-frequency form of the Fourier transform, known as Discrete Fourier transform (DFT), and the fundamental idea behind it is to sample the continuous-frequency discrete-time Fourier transform (DTFT) at at equal intervals of $\frac{2\pi}{N}$ over a period of 2π, be it from $[-\pi, \pi)$ or $[0, 2\pi)$, where N is the number of points in the sampled signal of interest. The DTFT is a frequency-normalized and repeating version of CTFT of the signal, i.e. $X(e^{j\omega}) = \frac{1}{T_s}\sum_{k=-\infty}^{\infty} X_c(j(\frac{\omega}{T_s} - \frac{2\pi k}{T_s}))$, where $T_s = \frac{1}{F_s}$ is the sampling interval, $\omega = \Omega T_s$ [rad] is the normalized angular frequency, and $X_c(j\Omega)$ is the CTFT of $x(t)$. Therefore, the DFT components for $0 \leq k < N$ can be written as $X[k] = X(e^{j(\frac{2\pi}{N})k}) = X(e^{j\omega})|_{\omega=(\frac{2\pi}{N})k}$, and zero for k everywhere else. In code, we have followed this procedure to construct the DFT $X[k]$ of the desired signal $x[n]$ consisting of two phasors at frequencies 440Hz and 660Hz and with a relative phase shift of $\frac{\pi}{2}$. Thus, the result in time domain, we already know, should be match $\hat{x}(t) = \cos(2\pi 440t) - \sin(2\pi 660t)$ sampled at $t = nT_s$, i.e. $\hat{x}[n] \coloneqq \hat{x}(nT_s)$. This has been verified and the frequency and time domain plots for $x[n]$ are shown in figure 1.

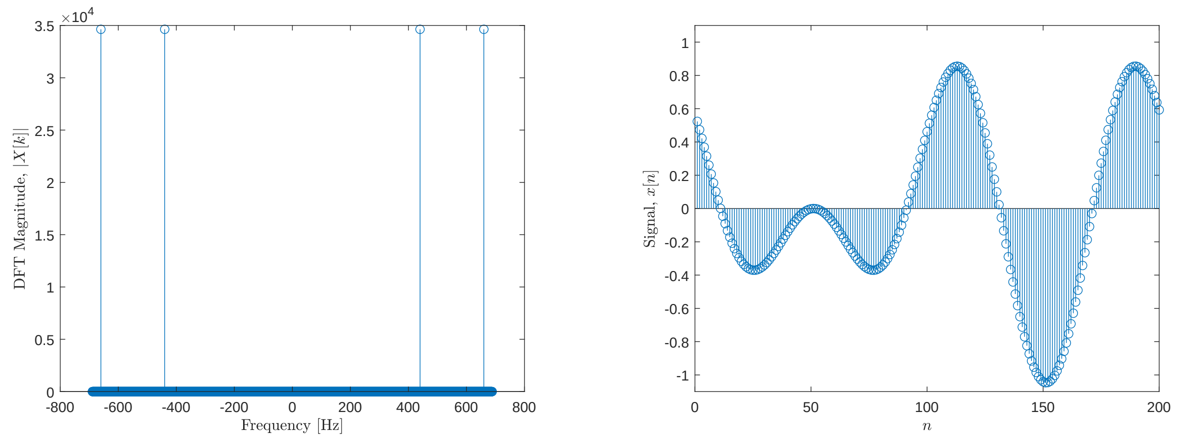
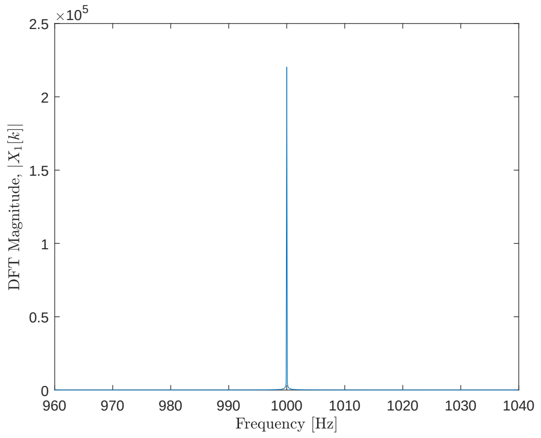


Figure 1: Frequency and time domain representations of $x[n]$

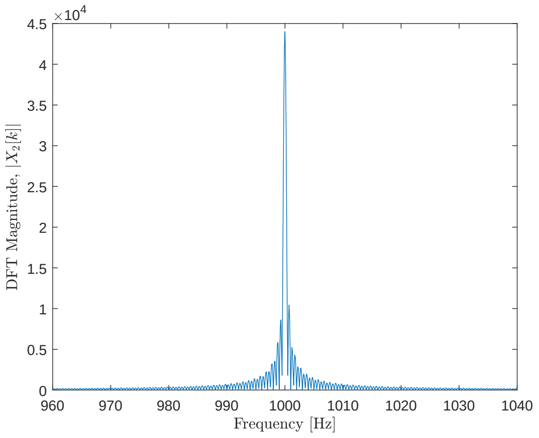
2 Effect of Time Windowing on the Frequency Spectra

We can practically only observe finite length signals, and this has a fundamental implication on how well we can localize the frequency information over the time which it happens. This is known as the time-frequency trade-off, which arises due to the convolution of the time-window spectra with the spectra of the infinite-length signal. Figures 2 (a) and (b) show the effect of relatively decreasing the observation time window of a sinusoidal signal, where we can clearly see that in (a) the energy is more sharply localized at frequency

$f_0 = 1000$Hz while (b) has larger spread (or distortion) around f_0 with a broader sinc function. As a limiting case, we would expect the positive spectra to converge to a δ function for an infinite-length time window. In general, this is expressed as the time-bandwidth relation where Δf is the observed bandwidth and Δt is the observation period, $\Delta f \Delta t \geq 1$.



(a) Spectra for $\Delta t = 10$s



(b) Spectra for (net) $\Delta t = 2$s

Figure 2: Effect of decreasing time window on frequency spectra

3 Telephone Dial Tones

Each dial tone signal was generated by adding two sinusoidal signals with the corresponding frequencies f_{lower} and f_{upper} listed in the DTMF (dual-tone multi-frequency) standard table.

4 Dial Tone Decode Algorithm

Given a DTMF signal, we would like to decode which key it corresponds to. This can be easily achieved once we know the f_{lower} and f_{upper} frequencies involved in the signal. Although this can be done using an analog circuit with tuned filters (in parallel) with a mediocre Q for making them robust to noise on the channel, we would like to develop the digital version of the process, an algorithm, that can be implemented, for example, on a microprocessor. Moreover, to make it robust to noise and distortion in the same way as the circuits, we introduce an $\epsilon-$frequency band around each DTMF frequency bin. The procedure for this is listed in algorithm 1, and the corresponding MATLAB code can be found in the appendix.

```
Algorithm 1 Robust DTMF Decoder
Require: Signal $x[n]$, sampling rate $F_s$, DTMF table $(L, U) \mapsto k$
  $\epsilon \leftarrow$ fr_eps
  $X \leftarrow \mathcal{F}\{x[n]\}$                              ▷ FFT of the signal
  for $\forall f_{l,i} \in f_{\text{lower}}$ and $\forall f_{u,j} \in f_{\text{upper}}$ do
      $m_{l,i} \leftarrow \sum_f ||X(|f - f_{l,i}| < \epsilon)||$
      $m_{u,j} \leftarrow \sum_f ||X(|f - f_{u,j}| < \epsilon)||$
  end for
  $L = \arg\max_i \quad [m_{l,i}]$                        ▷ Get index of the maximum
  $U = \arg\max_j \quad [m_{u,j}]$
  return DTMF$(L, U)$                         ▷ corresponding key $k$ is returned
```

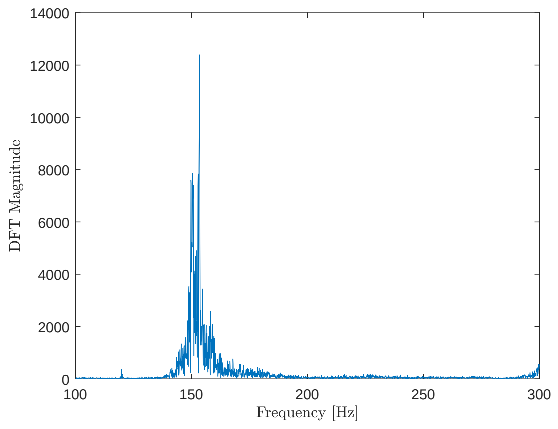
Figure 3: Algorithm for decoding DTMF tones to the corresponding key

5 Telephone Event Tones

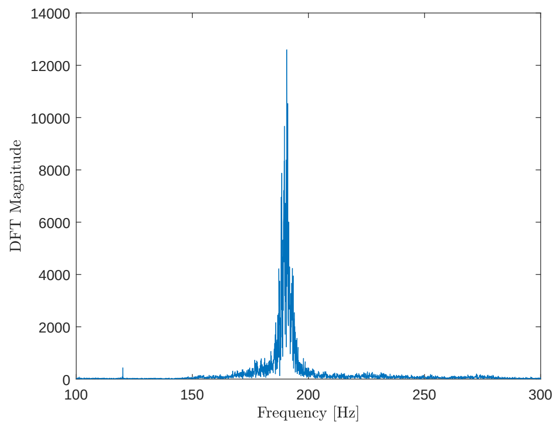
The tones for busy and continuous dial events were generated using the method described in section 3.

6 Helmholtz Resonators

The Helmholtz frequency for a resonator, $\omega_H = 2\pi f_H$, is given by $\omega_H = c\sqrt{\frac{S}{V_0 l}}$, where S is the cross-sectional area of the neck, l is the length of the neck, V_0 is the volume of the cavity or resonator, and $c = 343\text{m/s}$ is the speed of sound at 20°C. For a chosen bottle as a Helmholtz resonator, we were interested in finding its resonant frequency and its change with a decrease in available volume by a factor. After repeated trials, this factor was chosen to be $\frac{1}{3}$ instead of the suggested value of $\frac{1}{2}$, mainly since it was harder to achieve resonance for the half-filled bottle without creating excess noise caused by the blowing of air. Due to the volume change, we then had the cavity volume $V_1 = \frac{2}{3}V_0$ and we expected the observed resonant frequencies to shift by a factor of $\frac{\widehat{\omega_H}}{\omega_H} = \sqrt{\frac{V_0}{V_1}} = \sqrt{\frac{3}{2}}$. The chosen bottle had $V_0 = 750\text{ml}$, $l = 1\text{cm}$, and cross-sectional diameter $D = 2\text{cm}$. The resonant frequencies were observed as peaks on the FFT spectra shown in figure 4. For the empty bottle, we observed $f_h = 153.4\text{Hz}$, which is different from the expected value of $f_{H,\text{exp}} = 353.3\text{Hz}$. This could be because of the larger mouth of the bottle and the assumption for $V_0 \gg Sl$ not holding that well. However, the expected resonant frequency, $f_{H,\text{exp}} = 158\text{Hz}$, comes closer to f_H for $l = 5\text{cm}$. The observed frequency for the one-third filled bottle was $\widehat{f_H} = 190.7\text{Hz}$ which is close approximately equal to $f_H\sqrt{\frac{3}{2}} = 187.87\text{Hz}$. This confirms the inverse $\sqrt{V}$ relationship with the resonant frequency even if the apparent parameters for the resonator might be different than the measurements.



(a) f_H for empty bottle



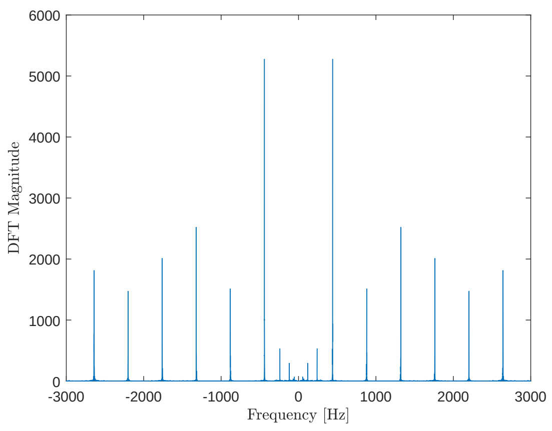
(b) f_H for one-third filled bottle

(c) Helmholtz resonator

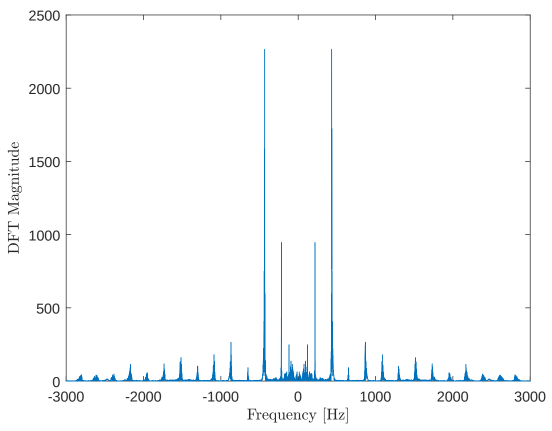
Figure 4: Observing Helmholtz resonance in a bottle

7 Comparison of Voice and Instrument Timbre

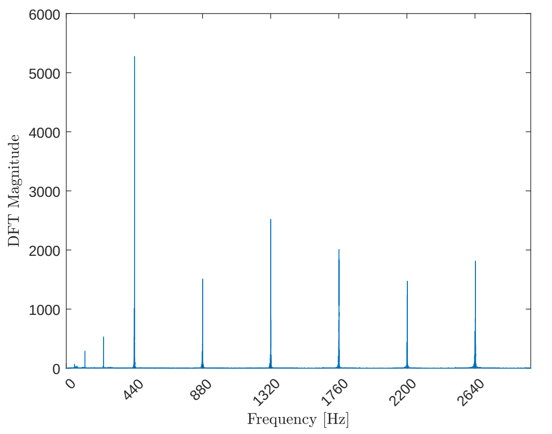
Here we compared the middle A note (at 440Hz) played by a violin and sung by a human (the author in this case). The notes in both cases sound quite different and that is attributed to the timbre of the instrument. Since the instrument has a complex body associated with it, on playing it, different modes are excited which give rise to a set of structured harmonics that collectively produce the note. The violin has 440Hz as the fundamental harmonic two sub-harmonics, and all along there is a structure to the spread of energy in the spectra. On the other hand, human voice recorded for this note has its main harmonic at 432Hz with a significant sub-harmonic at 216Hz and their multiples reduce in magnitude without much "structure". Additionally, the two signals are different in the time domain even though the dominant frequencies might be the same, specifically with the instrument having a characteristic intensity curve over time. The spectral comparison for the violin and human voice is shown in figure 5.



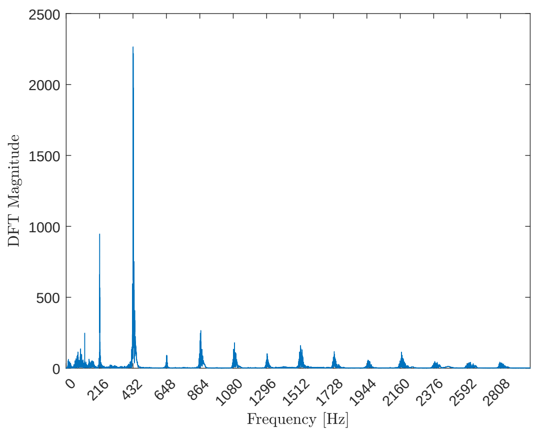
(a) Spectra for middle A (440Hz) on a violin



(b) Spectra for middle A (440Hz) in human voice



(c) Harmonics in the violin note

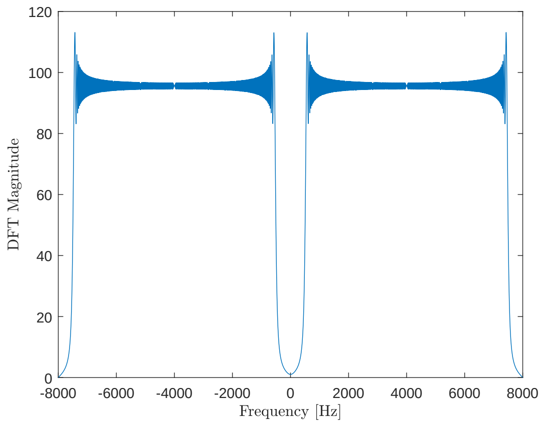


(d) Harmonics in the voice note

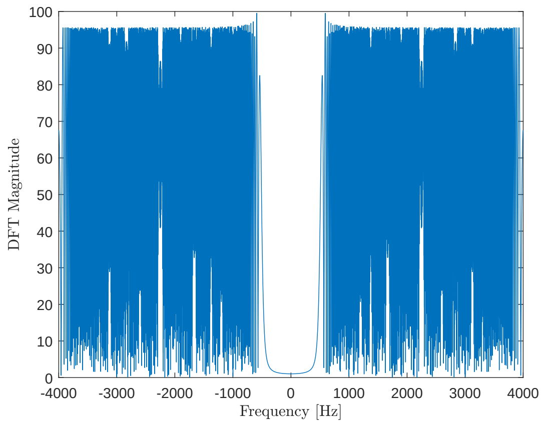
Figure 5: Comparing the spectra and harmonics in voice and instrument

8 Creating Chirps

A chirp is a signal whose (instantaneous) frequency changes with time, e.g. a constant amplitude chirp $x(t) = cos(2\pi\phi(t))$ where the instantaneous frequency, f, is given by $f(t) = \frac{d\phi}{dt}$. We considered a constant amplitude linear chirp starting at 500Hz and reaching 7.5kHz at $t = 1$s, which will have $\phi(t) = \frac{1}{2}7000t^2 + 500t$. Here we sampled such a chirp at $F_s = 16$kHz as well as at 8KHz which is below the Nyquist rate (15kHz in this case) and is expected to cause aliasing in which case, due to sub-sampling, the energy in the higher frequencies overlaps on lower frequency bands (in this case the whole spectra). This is confirmed in the frequency spectra for both signals shown in figure 6. Theoretically, the sampled signal can contain the maximum frequency of $\frac{F_s}{2}$ which is also what we observe in the spectra (and spectrograms in the next section).



(a) $F_s = 16$KHz

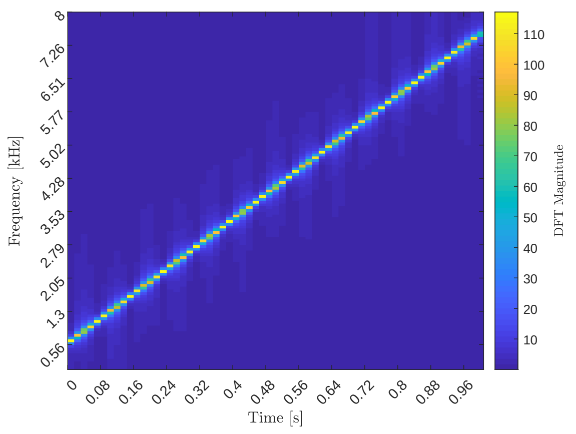


(b) $F_s = 8$KHz, aliased

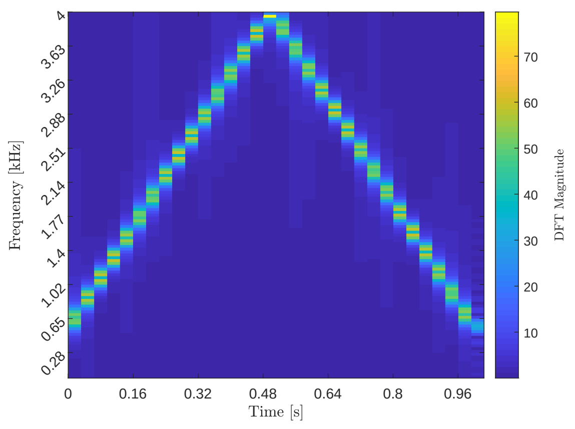
Figure 6: Linear chirp spectra and the effect of sub-sampling

9 Developing Simple Spectrograms

The spectrogram involved segmenting the time signal, potentially with overlaps, and then multiplying the segments with a windowing function after which an FFT for each segment and plotted over time range it signifies. A simple spectrogram can be made using rectangular windows with no overlaps, although this choice causes significant spectral leakage. But since we have crafted a signal (linear chirp from section 8) which has a single frequency at any given time, this choice for creating the spectrogram still produces good results that also match our expectation. In code, we have bundled this procedure in the custom `sonograph` function which is an alias of the `spectrogram` (built-in) function in MATLAB. The spectrograms for both chirp signals are shown in figure 7.



(a) Spectrogram for the linear chirp with $F_s = 16$kHz



(b) Spectrogram for the linear chirp with $F_s = 8$kHz

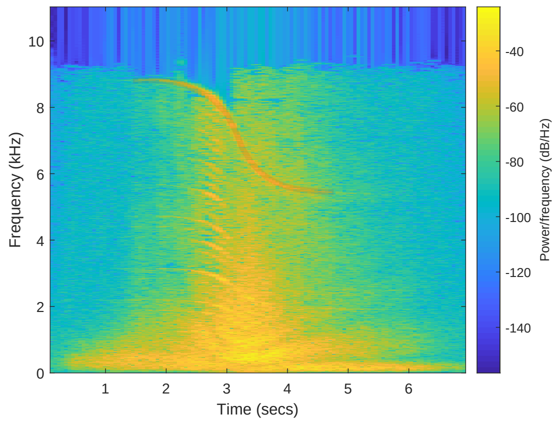
Figure 7: Spectrograms of linear chirps

10 Doppler Effect and Tracking Chirps on Spectrograms

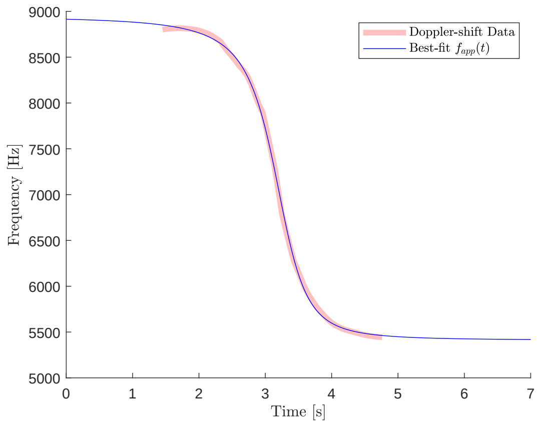
For an object moving with constant velocity, the apparent frequency $f_{\text{app}}(t)$ of a pure tone source as measured by a stationary observer is given by the expression,

$$f_{\text{app}}(t) = f_0 \frac{c}{c + v \sin\left(\arctan\left(\frac{v(t-t_0)}{d}\right)\right)},$$

where f_0 is the frequency of the source when it is stationary, v the velocity of the source, d the closest distance by which the source passes the microphone at t $= t_0$, and c is the speed of sound. We were given to deduce the $f_0, v,$ and d from a Learjet takeoff recording. The spectrogram highlighting the most prominent frequency shift along with the best-fit for $f_{\text{app}}(t)$ is shown in figure 8. The parameters for the best-fit, using non-linear regression, were found to be $f_0 = 6741.8$Hz, $v = 84.46$m/s $= 164.18$kn, $d = 40.43$m, and $t_0 = 3.29$s. However, there are major assumptions made while using this equation in our case; it is derived for a straight line trajectory of the source at a constant velocity, both of which will not hold strictly during takeoff and are only good to a basic approximation as the velocity deduced is still within 10% of the maximum takeoff velocity of 150kn for a Learjet.



(a) Spectrogram of observed jet takeoff sound



(b) Best-fit of the most significant "chirp"

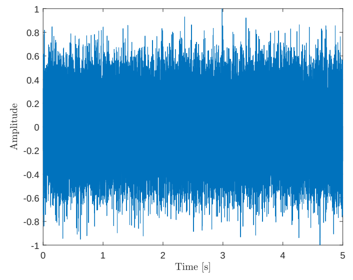
Figure 8: Doppler shift tracking and an estimate of $f_{\text{app}}(t)$

11 Generating Random Noise

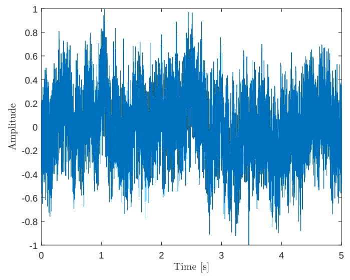
Color noise was generated using random phase with amplitude for each frequency bin $\propto 1/\sqrt{f^n}$. The corresponding time domain and power spectrum plots are contained in figures 9 and 10. In the time domain, we can see that the next value in white noise is not predictable from the previous value, while for pink and brown noise we increasingly see the traces memory due to the bandwidth limiting, or low-pass filtering, in the spectrum. With lower energy in the high frequency spectrum, the "colored" signal cannot transition to any arbitrary location on the next sample and instead it depends on previous samples. Therefore, the "depth" or extent of memory will increase in width depending on how narrow the filtering is in frequency which corresponds directly to the color of noise: red/brown for most energy in low frequencies, pink for mid-range, and white for equal energy over the entire spectrum.

12 Simpler Generation of Uniform Random (White) Noise

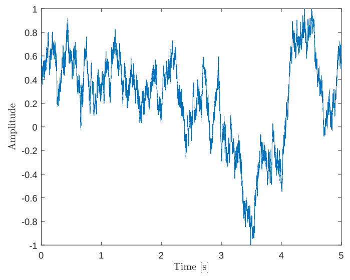
Given $U \sim \mathcal{U}(0,1)$, we can create uncorrelated uniform white noise with PCM encoding, i.e. $W \in [-1,1]$, by the transformation $W = 2U - 1$. A realization of the random process $W[n]$ is given in figure 11.



(a) White noise

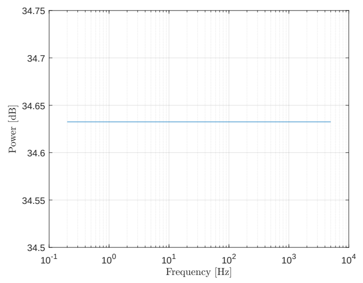


(b) Pink noise

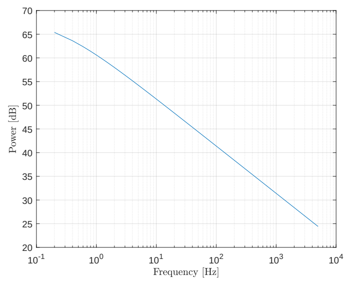


(c) Brown noise

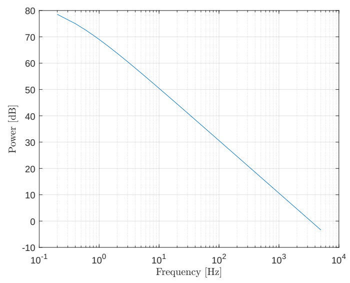
Figure 9: Color noise in time domain



(a) White noise



(b) Pink noise, slope −10dB/dec



(c) Brown noise, slope −20dB/dec

Figure 10: Power spectrum of color noise

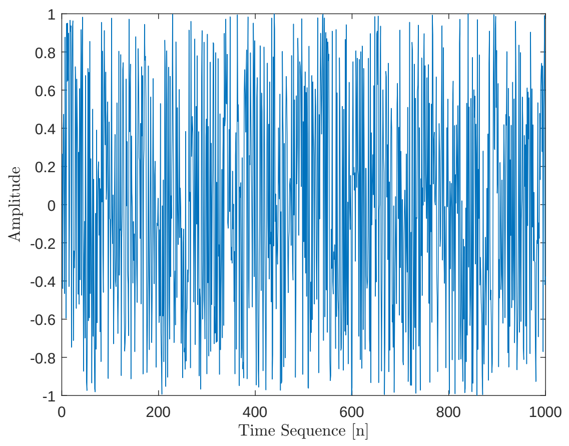


Figure 11: Uniform white noise sequence

Appendix

MATLAB code for algorithm 1 is as follows.

```
% simple dtmf decode algorithm for a subset of tones (can be extended)
input_dtmf_tone = '';
[signal, Fs]    = audioread(input_dtmf_tone);

N   = length(signal);   % number of points in input signal and its fft
df  = Fs/N;             % frequency increment in nyquist range
fr  = -Fs/2:df:Fs/2-df; % frequency range (nyquist range)

fr_eps      = 10;          % frequency window around pure tones
dtmf_matrix = [1 2; 4 5]; % (partial) dtmf decode matrix; indexed by lower and upper band ...
    freqs
signal_fft  = fftshift(fft(signal)); % fft of the input signal

% get magnitude of each tone's contribution to the signal spectrum
m_fl1 = sum(2*abs(signal_fft(abs(fr - fl1) < fr_eps)));
m_fl2 = sum(2*abs(signal_fft(abs(fr - fl2) < fr_eps)));
m_fu1 = sum(2*abs(signal_fft(abs(fr - fu1) < fr_eps)));
m_fu2 = sum(2*abs(signal_fft(abs(fr - fu2) < fr_eps)));

% get dtmf decoding matrix indexes
[¬, fl] = max([m_fl1 m_fl2]);
[¬, fh] = max([m_fu1 m_fu2]);

dtmf_tone = dtmf_matrix(fl, fh); % algorithm output
```