C.11 Sampling and Aliasing Solution

C.11.1 In-lab section

1. To get a frequency sweep from 0 to 12 kHz in 10 seconds we need to choose \( f \) so that \( 2fT = 12000 \) when \( t = 10 \). Thus, \( f = 600 \). The following Matlab code generates the chirp and plays it as a sound:

\[
\begin{align*}
t &= \{0:1/8000:10\}; \\
y &= \sin(2\pi 600 \times (t \times t)); \\
\text{soundsc}(y);
\end{align*}
\]

The perceived pitch rises from 0 to 4 kHz, then falls back to 0, then rises again to 4 kHz. The reason for this is that the sampled signal has frequency components whose frequencies are ambiguous, modulo \( f_s = 8000 \). The audio hardware in the computer is choosing to render only frequencies in the range \(-4\) kHz to 4 kHz.

2. To get a frequency sweep from 0 to 2500 Hz in 8192/8000 seconds we need to choose \( f \) so that \( 2fT = 2500 \) when \( t = 8192/8000 \). Thus,

\[
f = \frac{2500 \times 8000}{2 \times 8192} = 1221.
\]

The following Matlab code generates this chirp and plays the sound:

\[
\begin{align*}
D &= 8192/8000; \\
t &= \{0:1/8000:8191/8000\}; \\
f &= 1221; \\
y &= (1-\text{abs}(t-D/2)/(D/2)).*\sin(2\pi f \times (t \times t)); \\
\text{soundsc}(y);
\end{align*}
\]

There are no aliasing artifacts because the frequency remains below the Nyquist frequency.

3. The following Matlab code gives a correctly labeled plot of the DFT of the signal \( y \) calculated in the previous part:

\[
\begin{align*}
p &= \text{length}(y); \\
fs &= 8000; \\
Y &= \text{fft}(y); \\
Y_{\text{symmetric}} &= [Y(4097:8192), Y(1:4096)]; \\
symmetricFreqs &= [-fs/2:fs/p:(fs/2)-(fs/p)]; \\
\text{plot}(\text{symmetricFreqs}, \text{abs}(Y_{\text{symmetric}})); \\
xlabel(‘frequency in Hertz’); \\
ylabel(‘magnitude of DFT’);
\end{align*}
\]

The plot is shown in figure C.33. The plot is sensible, in that it shows the triangular shape that determines the relative amounts of each of the frequencies swept.
Figure C.33: Magnitude of the DFT of the chirp with a triangular envelope.
4. To get a frequency sweep from 0 to 5000 Hz in 8192/8000 seconds we need to choose \( f \) so that \( 2ft = 5000 \) when \( t = 8192/8000 \). Thus,

\[
f = \frac{5000 \times 8000}{2 \times 8192} = 2441.
\]

The following Matlab code generates this chirp and plays the sound:

```matlab
D = 8192/8000;
t = [0:1/8000:8191/8000];
f = 2441;
y = (1-abs((t-D/2)/(D/2))).*sin(2*pi*f*(t.*t));
soundsc(y);
```

There is now an aliasing artifact towards the end of the chip where the frequency drops rather than rises.

The same code as in the previous part can be used to plot the DFT, resulting in the plot shown in figure C.34. Notice that between 3 and 4 kHz there is aliasing distortion. This DFT is the result of overlapping and adding two DFTs that are shifted by 8 kHz in frequency. Since the sweep extends beyond the Nyquist frequency, there is non-zero overlap, resulting in the distortion shown.

5. We recreate the chirp from part 2, which sweeps from 0 to 2500 Hz, as follows:

```matlab
D = 8192/8000;
t = [0:1/8000:8191/8000];
f = 1221;
y = (1-abs((t-D/2)/(D/2))).*sin(2*pi*f*(t.*t));
```

To create the downsampling signal, one (clever) method is

\[
w = y([1:4096]*2);
\]

The following Matlab code gives a correctly labeled plot of the DFT:

```matlab
p = length(w);
fs = 4000;
W = fft(w);
Wsymmetric = [W(2049:4096), W(1:2048)];
symmetricFreqs = [-fs/2:fs/p:(fs/2)-(fs/p)];
plot(symmetricFreqs, abs(Wsymmetric));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

The plot is shown in figure C.35. It shows aliasing distortion because the new sample rate results in a Nyquist frequency that is below the highest frequency terms in the signal.
Figure C.34: Magnitude of the DFT of a chirp with a triangular envelope that sweeps beyond the Nyquist frequency.
Figure C.35: Magnitude of the DFT of the downsampled chirp.
6. Assuming the same signal $y$ constructed in the previous part, we can construct $z$ as follows:

```matlab
for(k=1:length(y))
    z(2*k-1) = y(k);
    z(2*k) = 0;
end
```

The following Matlab code gives a correctly labeled plot of the DFT:

```matlab
p = length(z);
fs = 16000;
Z = fft(z);
Zsymmetric = [Z(8193:16384), Z(1:8192)];
symmetricFreqs = [-fs/2:fs/p:(fs/2)-(fs/p)];
plot(symmetricFreqs, abs(Zsymmetric));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

The plot is shown in figure C.36.

7. To get back the original chirp, we could lowpass filter the signal with the following Butterworth filter:
\[ [B, A] = \text{butter}(10, 1/3); \]
\[ u = \text{filter}(B, A, z); \]

The result sounds just like the original chirp. You can plot its magnitude DFT as above, and it will look perfect. If you use a lower order filter, such as 4 instead of 10, then there will be small residuals in the frequency range 5500 to 8000 Hz.