

ECE 148 Homework 6

Sanjot Bains

May 21, 2025

I. Interpolation by DFT

Consider a real periodic signal $f(t)$ of period T . The signal has 7 harmonics, for $m = 1, 2, 3, \dots, 7$:

$$f(t) = \sum_{m=1}^7 a_m \sin(m\omega_0 t)$$

where $\omega_0 = \frac{2\pi}{T}$ and the 7 coefficients $\{a_m\}$ are formed with the 7-digit perm number: 9587189.

```
1 T = 1; % Period of the signal
2 w_0 = 2 * pi / T; % Fundamental frequency
3 a_m = [9 5 8 7 1 8 9]; % Amplitude of harmonics
4
5 f = @(t) a_m(1) * sin(1 * w_0 * t) + a_m(2) * sin(2 * w_0 * t) + ...
6         a_m(3) * sin(3 * w_0 * t) + a_m(4) * sin(4 * w_0 * t) + ...
7         a_m(5) * sin(5 * w_0 * t) + a_m(6) * sin(6 * w_0 * t) + ...
8         a_m(7) * sin(7 * w_0 * t);
```

Listing 1: Signal Definition

We take 16 uniform samples within one period with sample spacing $\Delta t = \frac{T}{16}$ to form a short 16-point sequence $\{f(n)\}$, where $n = 0, 1, 2, \dots, 15$.

```
1 delta_T = T / 16; % Sampling interval
2 t_samples = 0:delta_T:(T-delta_T); % Sampled time vector
3 f_n = f(t_samples); % Sampled function values {f(n = 0, 1, ..., 15)}
```

Listing 2: Sampling

Subsequently, we take a 16-point DFT of the sequence to obtain the 16-point spectral sequence $F(k)$, where $k = 0, 1, 2, \dots, 15$.

$$F(k) = DFT_{N=16}\{f(n)\}$$

```
1 F_k = fft(f_n); % F(k) = DFT of f(n = 0, 1, ..., 15)
2 F_k_shifted = fftshift(F_k);
```

Listing 3: 16-pt DFT

1. Interpolation of the DFT Spectrum

The sequence $f(n)$ is extended to 64 points by padding 48 zeros. The extended sequence $f_a(n)$ is in the form:

$$f_a(n) = \begin{cases} f(n) & n = 0, 1, 2, \dots, 15 \\ 0 & n = 16, 17, 18, \dots, 63 \end{cases}$$

```
1 f_a = zeros(1, 64);  
2 f_a(1:16) = f_n;
```

Listing 4: Zero-Padded Time Sequence

We then compute the DFT of the 64-pt sequence:

$$F_a(k) = DFT_{N=64}\{f_a(n)\}$$

```
1 F_a_shifted = fftshift(fft(f_a));
```

Listing 5: DFT of 64-pt Sequence

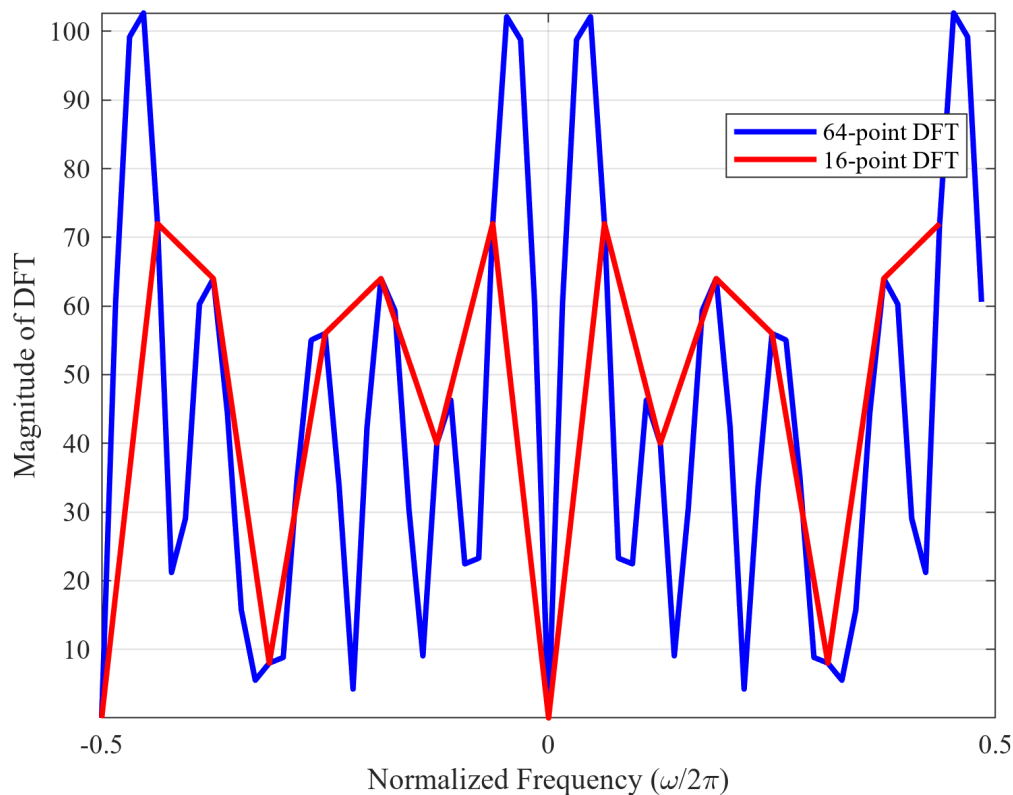


Figure 1: Comparison of 64-point and 16-point DFT

2. Interpolation in Time Domain

The 16-point spectral sequence $F(k)$ is extended to 64 points by inserting 48 zeros in the middle.

```
1 F_b = zeros(1, 64);
2 F_b(1:8) = F_k(1:8);
3 F_b(57:64) = F_k(9:16);
4 F_b = 4 * F_b;
```

Listing 6: Zero-Padded Frequency Sequence

The resulting interpolated time-domain signal $f_b(n)$ is computed using IDFT:

$$f_b(n) = IDFT_{N=64}\{F_b(k)\}$$

```
1 f_b = ifft(F_b);
```

Listing 7: IFFT

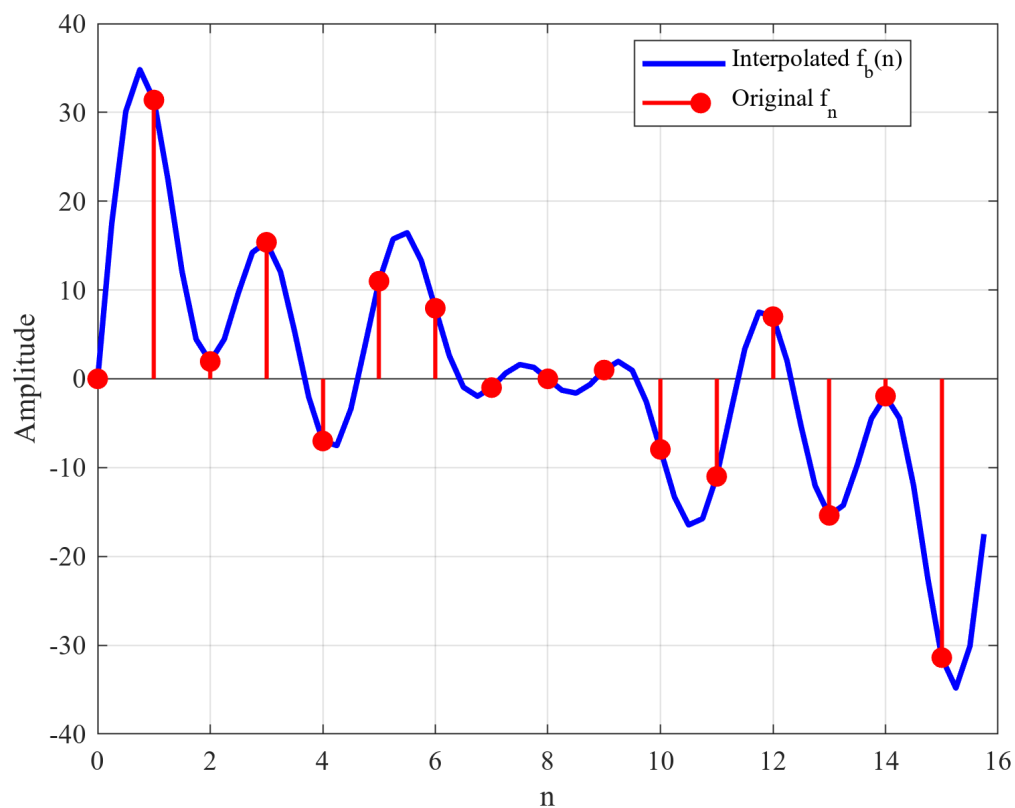


Figure 2: Comparison of Interpolated Signal with Original Samples

II. Signal Scrambling

The objective of this exercise is to implement a simple digital speech scrambler.

We use the microphone of our computer to record a short speech signal $g(t)$, and digitize the speech signal with the A/D tool in Audacity into the discrete form $g(n)$.

```
1 [g_n, f_s] = audioread('g_n.wav');
```

Listing 8: Reading in Audio File

1. DFT Spectrum

The DFT spectrum $G(k)$ of the digitized speech signal $g(n)$ is shown below:

```
1 G_k = fft(g_n);  
2 G_k_shifted = fftshift(G_k);
```

Listing 9: FFT of Audio Sequence

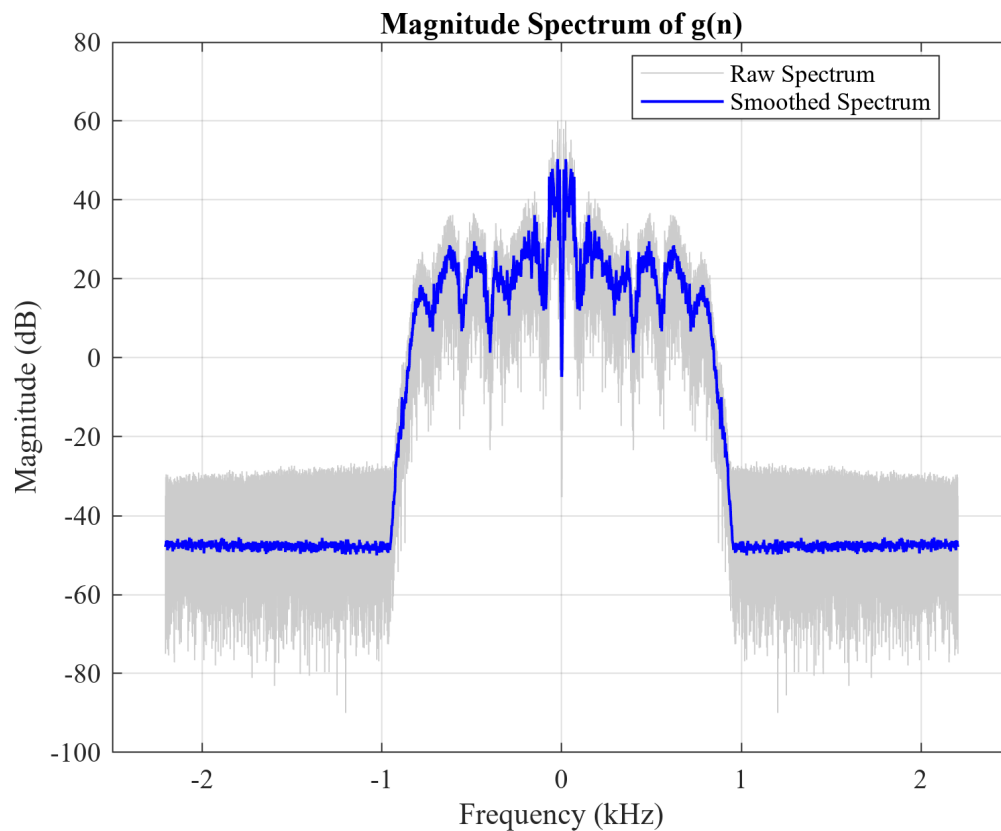


Figure 3: Magnitude Spectrum of Original Speech Signal

2. Speech Scrambling

The speech signal is scrambled by multiplying it with an alternating sequence of +1 and -1.

```
1 scrambling_seq = ones(size(g_n));  
2 scrambling_seq(2:2:end) = -1;  
3  
4 g_hat_n = g_n .* scrambling_seq;  
5  
6 G_hat_k = fft(g_hat_n);  
7 G_hat_k_shifted = fftshift(G_hat_k);
```

Listing 10: FFT of Hilbert Sequence

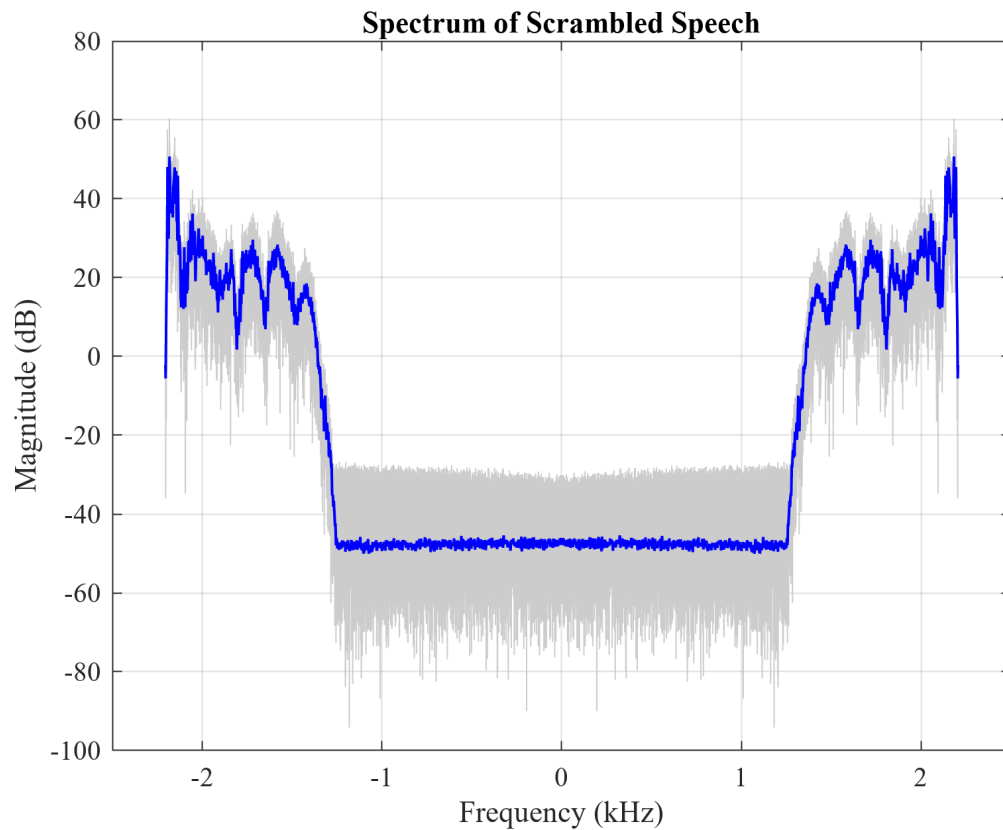


Figure 4: Spectrum of Scrambled Speech

3. Descrambling

The descrambling process uses the same alternating sequence. When there is a synchronization offset, it results in $-g(t)$ instead of $g(t)$.

```
1 descrambled_g_n = g_hat_n .* scrambling_seq;
2
3 audiowrite('descrambled_g_n.wav', descrambled_g_n, f_s);
```

Listing 11: Descrambling

III. Hilbert Transform

1. Periodic Signal $f(t)$

The Hilbert transform of the periodic signal $f(t)$ is computed and compared with the original signal:

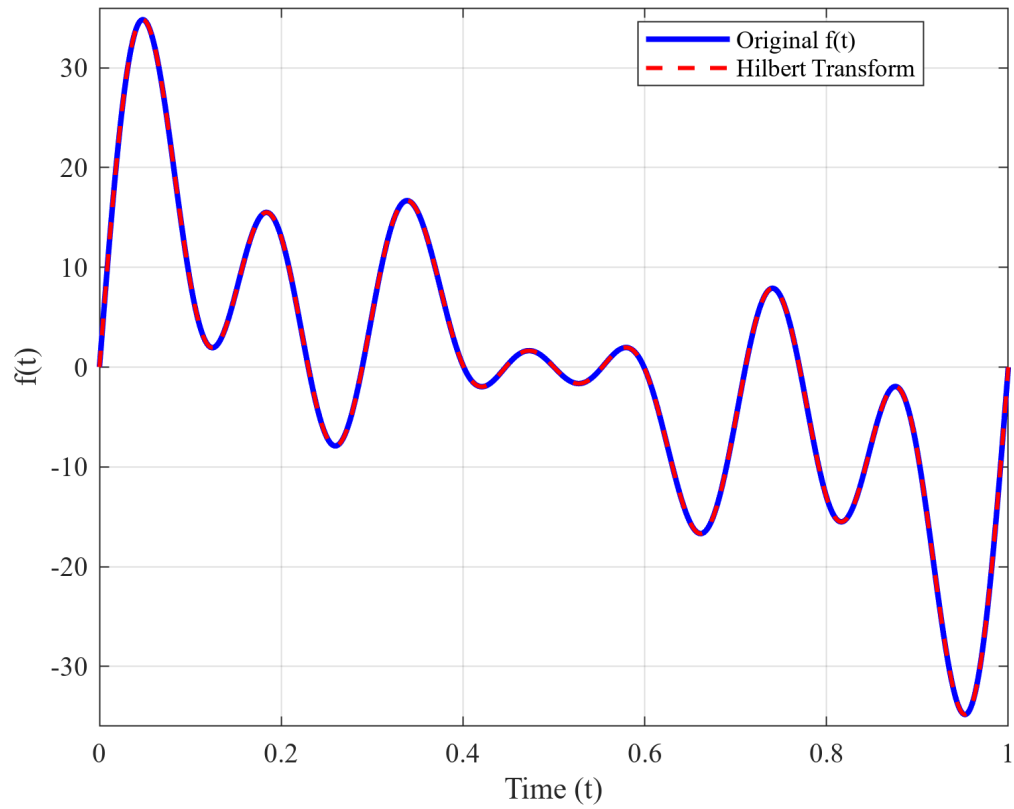


Figure 5: Original Signal and its Hilbert Transform

2. Speech Signal $g(t)$

The Hilbert transform is applied to the speech signal $g(t)$, and the result is saved as an audio file for audible comparison.