# ECE 146B Lab: Modulation, demodulation and synchronization in OFDM

**Lab Objectives:** To develop a hands-on understanding of basic OFDM transmission and reception, including key concepts related to synchronization.

**Laboratory Assignment**

Consider a discrete time complex baseband model in which a sequence of samples $\{s[n]\}$ sent by the transmitter give rise to a received signal

$$y[n] = (s * h)[n] + N[n]\ ,\ n = 1, ..., M \tag{1}$$

The non-trivial part of the channel impulse response is of length $L$ (the length is the difference between the earliest and latest nonzero entries in the impulse response, plus one), but we can model an additional delay of $n_0$ samples by inserting $n_0$ zeros at the beginning of the impulse response. For example, $h[n] = [0, 0, 1, 0, -0.5]$ is a channel of length $L = 3$, but the two zeros at the beginning model a delay of $\tau = 2$ samples.

We will be considering an OFDM system with $N$ subcarriers and a cyclic prefix of $P$. In this lab, you will write code for general $N$ and $P$, and demonstrate that the same code works for different values of $N$ and $P$. Specifically, you will consider two example settings in this lab: (1) $N = 16$, $P = 4$ (helpful for debugging), (2) $N = 256$, $P = 16$ (to check that your program works for a larger set of subcarriers).

The receiver local oscillator (LO) is not *a priori* synchronized with the transmitter's LO, and there may be a Doppler shift due to relative motion between the transmitter and receiver. In discrete time, this may be modeled as a phase offset of $2\pi\delta/N$ per sample, where $\delta$ is the carrier offset *normalized to the subcarrier spacing of* $1/N$, as follows:

$$y[n] = e^{j2\pi\delta n/N}(s * h)[n] + N[n]\ ,\ n = 1, ..., M \tag{2}$$

## 1 Part 1: Timing and Frequency Synchronization

**Step 1 (OFDM modulation and time domain statistics):** Write code implementing the basic OFDM modulator as follows.
**1a)** Generate $N$ Gray coded QPSK symbols $\mathbf{B} = \{B[k], k = 1, .., N\}$. Take the inverse FFT to obtain time domain samples $\mathbf{b} = \{b[n], n = 1, ..., N\}$.
*Remark:* Matlab's *ifft* function implements:

$$b[n] = \frac{1}{N}\sum_{k=1}^{N} B[k]e^{j2\pi(n-1)(k-1)} \tag{3}$$

but if we want to normalize the complex exponential basis functions to unit norm, we could multiply this by $\sqrt{N}$.
**1b)** Append the last $P$ time domain samples to the beginning, to get a length $N + P$ sequence of time domain samples $\mathbf{s} = \{s[n], n = 1, ..., N + P\}$. That is, $s[1] = b[N - P + 1], ..., s[P] = b[N], s[P + 1] = b[1], ..., s[N + P] = b[N]$.
**1c)** Plot histograms of the real and imaginary parts of $\mathbf{s}$. Do they look Gaussian?
**1d)** Compute the normalized correlation of the real and imaginary parts of $\mathbf{s}$. Is it reasonable to model them as uncorrelated?
**1e)** Compute the peak-to-average power ratio of $\mathbf{s}$:

$$PAR = \frac{\max_n |s[n]|^2}{\overline{|s[n]|^2}} \tag{4}$$

For BER simulations, you can use the function qpskmap developed in Software Lab 6.1 for this purpose. But for the development on synchronization in the first few parts, we can simply generate random $\{\pm 1 \pm j\}$ without keeping track of the bit-to-symbol map.

Plot histograms of $PAR(dB) = 10\log_{10}(PAR)$ over multiple OFDM symbols generated as in 1(a)-(b) for $N = 16$ ($P = 4$) and for $N = 256$ ($P = 20$). Comment on the dependence on $N$.

**Step 2 (Modeling and estimating delay)** Even if the channel is ideal, the receiver does not know where the OFDM symbol starts. To model this, consider a channel

$$h = [zeros(\tau, 1); 1], \quad \textbf{pure delay} \tag{5}$$

where $\tau \geq 0$ is an integer. Set the frequency offset $\delta = 0$. Now, in (1), we obtain a signal $y$ of length $N + P + \tau$, where the first $\tau$ entries are only noise. We would like to throw these entries away and then strip away the cyclic prefix. We do this by looking for a match between two segments of $P$ samples spaced $N$ samples apart, to estimate where the cyclic prefix is occurring. Compute the correlations between such segments for different hypothesized values of $\tau$:

$$R[k] = \sum_{n=k+1}^{k+P} y^*[n]y[n+N] \ , k = 0, 1, ..., K \tag{6}$$

$K + N + P = length(y)$. We can now estimate the best integer delay as

$$\hat{\tau} = \arg\ \max_{1\leq k\leq K}|R[k]| \tag{7}$$

After doing this, we can strip away the cyclic prefix and get the following time domain received signal:

$$\hat{y}[n] = y[n + \hat{\tau} + P], \ n = 1, ..., N \tag{8}$$

The $N$-length received signal $\mathbf{Y}$ in the frequency domain is now given by

$$Y[k] = FFT(\hat{y}), \ k = 1, ..., N \tag{9}$$

*Remark:* Matlab's *fft* function implements

$$Y[k] = \sum_{n=1}^{N} \hat{y}[n]e^{-j2\pi(n-1)(k-1)} \ , \ k = 1, ..., N \tag{10}$$

You may multiply this by $1/\sqrt{N}$ if you want to normalize the complex exponential basis functions to unit norm.
**2a)** Implement the preceding procedures for estimating $\hat{\tau}$ and computing $\mathbf{Y} = (Y[1], ..., Y[N])^T$. Use $N = 16$, $P = 4$ to evaluate your code in the following.
**2b)** For a noiseless system with $\tau = 3$ check that you obtain $\hat{\tau} = 3$. Does a scatterplot of $\{Y[k], k = 1, ..., N\}$ correspond to a QPSK constellation.
**2c)** Now, suppose that $\tau = 3$. Manually set $\hat{\tau} = 5$ in (8) and display a scatterplot of $\{Y[k], k = 1, ..., N\}$. What does it look like? Can you explain what you are seeing?

**3) Step 3 (Estimating and undoing small frequency offsets)** Assume that the normalized frequency offset $\delta$ in (2) is in $(-0.5, 0.5)$. Set $N = 16$, $P = 4$, since we are still in debug mode.
**3a)** Argue that

$$\widehat{\delta} = \angle R[\hat{\tau}] \tag{11}$$

the angle of $R[\hat{\tau}]$ provides an estimate for $\delta \in (-0.5, 0.5)$. Would this estimate work for larger values of $\delta$ (e.g., $\delta = 5.25$)?
**3b)** Consider a noiseless system with $\delta = 0.25$. Compare the estimate $\widehat{\delta}$ with the true value of $\delta$.
**3c)** Undo the frequency offset using your estimate. That is, replace $y[n]$ by $y[n]e^{-j2\pi\delta n/N}$ and then strip away the cyclic prefix and compute the length $N$ symbol $\mathbf{Y}$ as in (8)-9).

**3c** What frequency offset in parts per million (ppm) does $\delta = 0.25$ correspond to for a carrier frequency of 2.4 GHz, a bandwidth of 20 MHz and $N = 16$ subcarriers? How does your answer change for a carrier of 28 GHz, a bandwidth of 100 MHz and $N = 4096$ subcarriers?

**4) Step 4 (adding in a non-trivial channel)** Now, consider a simple two-tap channel (together with a delay $\tau$)

$$h = [zeros(\tau, 1); 1; -0.5], \quad \textbf{two} - \textbf{tap channel} \tag{12}$$

We are now going to check that the methods of Step 2 and Step 3 still apply for getting coarse timing synchronization and an estimate of normalized frequency offset in $(-0.5.0.5)$. This is because the periodicity in the transmitted stream induced by the cyclic prefix still persists when we pass it through an LTI system.

**4a)** For a noiseless system, redo Step 2 (with zero frequency offset) and show that you can still get a good estimate of $\tau$.

**4b** For a system with normalized frequency offset $\delta = 0.25$, redo Step 4 and show that you can still get a good estimate of $\delta$.

**4c)** Undo the frequency offset using your estimate. That is, replace $y[n]$ by $y[n]e^{-j2\pi\delta n/N}$ and then strip away the cyclic prefix and compute the length $N$ symbol $\mathbf{Y}$ as in (8)-9).

**Summary:** At this point, your code operates should be able to estimate and correct for a bulk delay $\tau$ and a small normalized frequency offset $\delta \in (-0.5, 0.5)$ for an OFDM symbol, ending up with a $N \times 1$ frequency domain observation $\mathbf{Y}$.

**Now we need to estimate the channel:** We have simply used the structure of the OFDM symbol in Steps 2-4 above. That is, we can do coarse timing synchronization and carrier synchronization (if the frequency offset is small enough) without needing a training sequence. We can do this even if in the presence of a nontrivial channel. However, demodulating the transmitted symbols requires that we obtain an estimate of frequency domain channel coefficients $\{H[k]\}$ in a model of the form

$$Y[k] = H[k]B[k] + \text{noise} \ , \ k = 1, ..., N \tag{13}$$

applied to the received vector obtained in (9). Once we obtain estimates $\{\widehat{H}[k]\}$ of the frequency domain channel, the symbols can be estimated by dividing by the channel estimated for each subcarrier (with adjustments to protect against dividing by small values):

$$\widehat{B}[k] = \frac{Y[k]}{\widehat{H}[k]} \ , \ k = 1, ..., N \tag{14}$$

In Part 2 of the lab, we provide some background on least squares channel estimation, and then ask you to implement and test it.

## 2 Part 2: Least Squares Channel Estimation

For channels that vary slowly, we could send one OFDM symbol which is completely known, and use the associated channel estimates for a large number of subsequent symbols. For channels that vary fast, we often use comb-style pilots, in which a subset of subcarriers in each symbol is set aside as pilot symbols. While we do not model channel time variations here, let us explore how to estimate the channel coefficients $\{H[k]\}$ when we know the symbols transmitted on a selected set of subcarriers.

When we need to learn $N$ frequency domain channel coefficients $\{H[k]\}$, these depend on an $L$-length time domain channel $\mathbf{h} = (h[0], ..., h[L-1])^T$, where $L \leq P \ll N$ in typical designs. Thus, we have fewer parameters to estimate if we work in the time domain. Let us assume that we know the symbols sent on a subset $\mathcal{T}$ of subcarriers. In particular, let us assume that we know the symbols on subcarriers spaced by $S$: $B[k], k \in \mathcal{T} = \{\infty, \infty + \mathcal{S}, \infty + \in\mathcal{S}, ...\}$ are known. The number of pilot, or training, symbols $T \approx N/(1+S)$. If $T > L$, the number of unknown taps, we hope to have enough equations to solve for the time domain channel, and from there, compute the frequency domain coefficients $\{H[k]\}$ via the FFT.

In least squares estimation, we seek to find $\mathbf{h}$ so as to minimize the squared error:

$$\sum_{k \in \mathcal{T}} |Y[k] - H[k]B[k]|^2 \tag{15}$$

where

$$H[k] = \sum_{l=0}^{L-1} h[l]e^{-j2\pi kl/N} \ , k \in \mathcal{T}$$

depends linearly on the $L \times 1$ time domain channel $\mathbf{h}$. The mathematical machinery to accomplish this is similar to that used for MMSE interference suppression, and is easiest to use when the problem is set up in matrix-vector format.

We wish to set up a matrix-vector formulation to make it easy to set up and solve a least squares problem. To this end, define $\mathbf{Y}_T$ as a $T \times 1$ column vector containing $Y[k], k \in \mathcal{T}$, $\mathbf{B}_T$ as a $T \times 1$ column vector containing $B[k], k \in \mathcal{T}$, and $\mathbf{H}_T$ as a $T \times 1$ column vector containing $H[k], k \in \mathcal{T}$ Define the $T \times L$ FFT matrix $\mathbf{F}_T$ with entries $\exp(-j2\pi kl/N)$, $k \in \mathcal{T}$, $l = 0, 1, ..., L-1$, so that

$$\mathbf{H}_T = \mathbf{F}_T \mathbf{h} \tag{16}$$

We are trying to fit $\mathbf{Y}_T$ to $\mathbf{B}_T \odot \mathbf{H}_T$, where $\odot$ denotes element-wise product. We can accomplish this by element-wise product operation by multiplying $\mathbf{H}_T$ by $diag(\mathbf{B}_T)$, a diagonal matrix with diagonal entries drawn from $\mathbf{B}_T$. We therefore have that

$$\mathbf{B}_T \odot \mathbf{H}_T = \mathbf{B}_T \odot (\mathbf{F}_T \mathbf{h}) = diag(\mathbf{B}_T)\mathbf{F}_T \mathbf{h} = \mathbf{A}\mathbf{h} \tag{17}$$

where

$$\mathbf{A} = diag(\mathbf{B}_T)\mathbf{F}_T \tag{18}$$

The squared error (15) we wish to minimize can now be written compactly as

$$J(\mathbf{h}) = ||\mathbf{Z} - \mathbf{A}\mathbf{h}||^2 = (\mathbf{Z} - \mathbf{A}\mathbf{h})^H (\mathbf{Z} - \mathbf{A}\mathbf{h}) \tag{19}$$

**Step 5 (Derivation of least squares channel estimate)**
**5a)** Show that the gradient of $J(\mathbf{h})$ with respect to $\mathbf{h}^*$ is given by

$$\nabla_{\mathbf{h}^*} J(\mathbf{h}) = -\mathbf{A}^H (\mathbf{Z} - \mathbf{A}\mathbf{h}) \tag{20}$$

**5b)** Conclude that the least squares channel estimate is given by

$$\widehat{\mathbf{h}} = \left(\mathbf{A}^H \mathbf{A}\right)^{-1} \mathbf{A}^H \mathbf{Z} \tag{21}$$

**5c)** Show that the minimum value of the squared error obtained by the solution in (21) is given by

$$J(\widehat{\mathbf{h}}) = \mathbf{Z}^H \mathbf{Z} - \mathbf{Z}^H \mathbf{A}\widehat{\mathbf{h}} \tag{22}$$

After we obtain this estimate, we can compute the frequency domain channel coefficients $\{\widehat{H}[k], k = 0, 1, ..., N-1\}$ by computing an $N$-point FFT on the $L$-dimensional estimate $\widehat{\mathbf{h}}$:

$$\widehat{\mathbf{H}} = fft(\mathbf{h}, N) \tag{23}$$

We can now do frequency domain equalization as in (14) to obtain symbol estimates $\{\widehat{B}[k]\}$.

**Step 6 (First pass implementation of least squares channel estimate)**
**6a)** Starting from the end of Step 4(c), implement the least squares estimate (21) assuming that you know the pilot symbols $B[k]$ for $B[k], k \in \mathcal{T} = \{\infty, \infty + \mathcal{S}, \infty + \in\mathcal{S}, ...\}$. Keep the value of $S$ (and all the other parameters) programmable. Run your code for debugging for $N = 16$, $P = 4$, $S = 2$, $h = (0; 0; 1, -0.5)$, and nominal channel length $L = P = 4$. You may try out your code first for $\delta = 0$ and then for $\delta = 0.25$.
**6b)** Comment on your results. From the coarse timing synchronization, you should have gotten $\hat{\tau} = 2$, and upon stripping away the cyclic prefix, the ideal value of the length 4 channel estimate would be $\mathbf{h} = (1, -0.5, 0, 0)^T$. How does the estimated length 4 channel compare with this ideal? Display a scatterplot of the estimated symbols $\{\widehat{B}[k]\}$. Does it look like a QPSK constellation?

**A potential complication:** The preceding approach should work in the debug setting considered in 6(a)-(b), but there could be difficulties when the channel is more challenging, even without noise. Specifically, if the initial estimate $\hat{\tau}$ used to strip away the cyclic prefix is different from the actual delay $\tau$, the channel coefficients in the frequency domain model (13) correspond to a cyclic shift (modulo $N$) of the $L$-length channel $\mathbf{h}$. This cyclic shift would typically be small, since we expect the error $\Delta\tau = \hat{\tau} - \tau$ to be small, but it would mean that the model (23) would no longer work. In order to address this, we can hypothesize different values of the offset $\Delta\tau$ when trying to fit the model. Specifically, a cyclic shift of $\Delta\tau$ corresponds to a phase offset of $e^{-j2\pi k\Delta\tau/N}$ in the $k$th subcarrier. We can now augment the procedure in Step 6 to account for different possible values of $\Delta\tau$, and choose the best, as follows.

For a hypothesized $\Delta\tau$, the channel coefficients in the frequency domain corresponding to $\mathbf{h}$ are given by

$$H[k] = fft(\mathbf{h}, N) \odot \mathbf{v}(\Delta\tau) \tag{24}$$

where $\mathbf{v}(\Delta\tau)[k] = e^{-j2\pi k\Delta\tau/N}$, $0 \leq k \leq N-1$ is the vector of phase shifts corresponding to $\Delta\tau$. The formulation of least squares channel estimation can be applied to estimate $\mathbf{h}$ for every hypothesized $\Delta\tau$ by modifying $\mathbf{F}_T$, and hence $\mathbf{A}$.

**Step 7 (Final implementation of least squares channel estimate, accounting for possible cyclic shift)**
**7a)** Implement least squares channel estimation for a set of hypothesized values of $\Delta\tau$ (e.g, $-2 \leq \Delta\tau \leq 2$) and pick the one that gives the smallest squared error.
**7b)** Run your code in debug mode in the setting of Steps 6(a)-(b), and report on the channel estimate and demodulated constellation that you obtain. You will probably find that your algorithm picks $\Delta\tau = 0$ as the best setting.
**7c)** Now, run your code for $N = 256$, $P = 16$, $S = 8$ and channel $\mathbf{h} = (0, 0, 0, 0, 2, -0.5j, 0.8j, 0, 0, -1, -j)^T$. What is the value of $\hat{\tau}$ that you obtain from coarse synchronization, and how does it compare with the actual delay $\tau = 4$? What is the value of $\Delta\tau$ that your algorithm gives? Do you get a QPSK-like scatterplot after frequency domain equalization (14)?

## 3 Part 3: Putting it all together

We now introduce noise and multiple OFDM symbols.

**Step 8 (adding noise)**
**8a)** Going back to (2), modify your code to add i.i.d. $CN(0, 2\sigma^2)$ noise samples $N[n]$ in the time domain, choosing $\sigma^2$ as a function of a specified $E_b/N_0$. Use an analytical estimate of $E_b$ accounting for the cyclic prefix, the scaling you have adopted in the IFFT at the transmitter, and the channel, but also check it against simulation, where $E_b$ is computed by taking the energy of the noiseless received signal and dividing it by the number of bits in the payload. For this computation, you may assume that the pilot symbols are also part of the payload.
**8b)** For a large $E_b/N_0$ of 20 dB, run your code for coarse synchronization, offset correction, channel estimation and demodulation, and check via a scatterplot that the equalized symbols in the two system settings we have considered look like a QPSK constellation.

**Step 9 (BER simulations)**
Now, think of the first symbol as a synchronization symbol, and concatenate 10 OFDM symbols carrying Gray coded QPSK payload on all subcarriers. Use the second system setting (larger $N$, more complicated channel). Allow the $E_b/N_0(sync)$ for the first synchronization symbol to be different from $E_b/N_0(payload)$ values for the payload symbols. Estimate the BER obtained by the system (using multiple runs), and plot it (on a log scale) against $E_b/N_0(payload)$ (dB) (over a range 5-20 dB, say). Plot three curves: (1) ideal QPSK, (2) Your system with $E_b/N_0(sync)$ of 30 dB, which should yield near-noiseless performance (3) Your system with $E_b/N_0(sync)$ of 12 dB. Comment on the relation between the curves.

**Lab Report:** Your lab report should document the results of the preceding steps in order. Describe the reasoning you used and the difficulties you encountered.