Orthogonal Frequency Division Multiplexing (OFDM): Tutorial and Analysis

December 11, 2001

Erich Cosby

ecosby@vt.edu

Virginia Tech.
Northern Virginia Center

1 INTRODUCTION

1.1 Purpose

Efficient use of radio spectrum includes placing modulated carriers as close as possible without causing Inter-Carrier Interference (ICI). Optimally, the bandwidth of each carrier would be adjacent to its neighbors, so there would be no wasted spectrum. In practice, a guard band must be placed between each carrier bandwidth to provide a space where a filter can attenuate an adjacent carrier's signal. These guard bands are wasted bandwidth.

In order to transmit high data rates, short symbol periods must be used. The symbol period is the inverse of the baseband data rate (T = 1/R), so as R increases, T must decrease. In a multi-path environment, a shorter symbol period leads to a greater chance for Inter-Symbol Interference (ISI). This occurs when a delayed version of symbol 'n' arrives during the processing period of symbol 'n+1'.

Orthogonal Frequency Division Multiplexing (OFDM) addresses both of these problems. OFDM provides a technique allowing the bandwidths of modulated carriers to overlap without interference (no ICI). It also provides a high date rate with a long symbol duration, thus helping to eliminate ISI. OFDM may therefore be considered as a candidate modulation technique in a broadband, multi-path environment.

The purpose of this report is to provide the following information concerning OFDM:

- theory of operation
- analysis of important characteristics
- implementation example (matlab)

1.2 OFDM Overview

OFDM is a modulation technique where multiple low data rate carriers are combined by a transmitter to form a composite high data rate transmission. Digital signal processing makes OFDM possible. To implement the multiple carrier scheme using a bank of parallel modulators would not be very efficient in analog hardware. However, in the digital domain, multi-carrier modulation can be done efficiently with currently available DSP hardware and software. Not only can it be done, but it can also be made very flexible and programmable. This allows OFDM to make maximum use of available bandwidth and to be able to adapt to changing system requirements.

Each carrier in an OFDM system is a sinusoid with a frequency that is an integer multiple of a base or fundamental sinusoid frequency. Therefore, each carrier is like a Fourier series component of the composite signal. In fact, it will be shown later that an OFDM signal is created in the frequency domain, and then transformed into the time domain via the Discrete Fourier Transform (DFT).

Two periodic signals are *orthogonal* when the integral of their product, over one period, is equal to zero. This is true of certain sinusoids as illustrated in Equation 1.

Equation 1 : Definition of Orthogonal

Continuous Time :

$$\int_0^T \cos(2\pi n f_0 t) \times \cos(2\pi m f_0 t) dt = 0 \quad (n \neq m)$$

Discrete Time :

$$\sum_{k=0}^{N-1} \cos\left(\frac{2\pi k n}{N}\right) \times \cos\left(\frac{2\pi k m}{N}\right) = 0 \quad (n \neq m)$$

The carriers of an OFDM system are sinusoids that meet this requirement because each one is a multiple of a fundamental frequency. Each one has an integer number of cycles in the fundamental period.

2 OFDM OPERATION

2.1 Preliminary Concepts

When the DFT (Discrete Fourier Transform) of a time signal is taken, the frequency domain results are a function of the *time sampling period* and the *number of samples* as shown in Figure 1. The fundamental frequency of the DFT is equal to 1/NT (1/total sample time). Each frequency represented in the DFT is an integer multiple of the fundamental frequency. The maximum frequency that can be represented by a time signal sampled at rate 1/T is f_{max} = 1/2T as given by the Nyquist sampling theorem. This frequency is located in the center of the DFT points. All frequencies beyond that point are images of the representative frequencies. The maximum frequency bin of the DFT is equal to the sampling frequency (1/T) minus one fundamental (1/NT).

The IDFT (Inverse Discrete Fourier Transform) performs the opposite operation to the DFT. It takes a signal defined by frequency components and converts them to a time signal. The parameter mapping is the same as for the DFT. The time duration of the IDFT time signal is equal to the number of DFT bins (N) times the sampling period (T).

It is perfectly valid to generate a signal in the frequency domain, and convert it to a time domain equivalent for practical use*. This is how modulation is applied in OFDM.

> * The frequency domain is a mathematical tool used for analysis. Anything usable by the real world must be converted into a real, time domain signal.

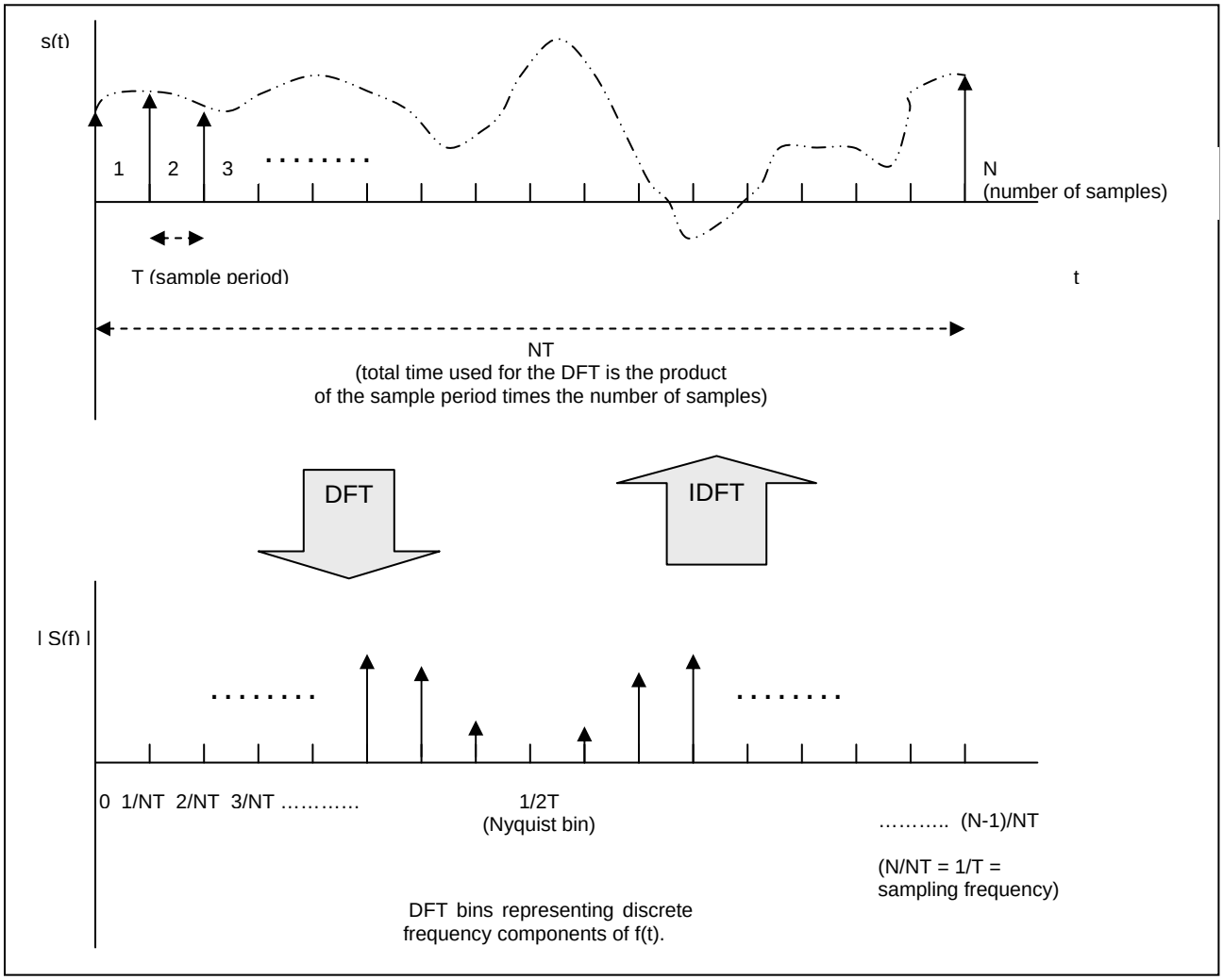

Figure 1: Parameter Mapping from Time to Frequency for the DFT

In practice the Fast Fourier Transform (FFT) and IFFT are used in place of the DFT and IDFT, so all further references will be to FFT and IFFT.

2.2 Definition of Carriers

The maximum number of carriers used by OFDM is limited by the size of the IFFT. This is determined as follows in Equation 2:

Equation 2 : OFDM Carrier Count

$$N_{carriers} \leq \frac{IFFTsize}{2} - 2 \quad \text{(real - valued time signal)}$$

$$N_{carriers} \leq IFFTsize - 1 \quad \text{(complex - valued time signal)}$$

In order to generate a real-valued time signal, OFDM (frequency) carriers must be defined in complex conjugate pairs, which are symmetric about the Nyquist frequency (f_{max}). This puts the number of potential carriers equal to the IFFT size/2. The Nyquist frequency is the symmetry point, so it cannot be part of a complex conjugate pair. The DC component also has no complex conjugate. These two points cannot be used as carriers so they are subtracted from the total available.

If the carriers are not defined in conjugate pairs, then the IFFT will result in a time domain signal that has imaginary components. This must be a viable option as there are OFDM systems defined with carrier counts that exceed the limit for real-valued time signals given in Equation 2. Reference [1] describes a system with IFFT size 256 and carrier count 216. This design must result in a complex time waveform. Further processing would require some sort of quadrature technique (use of parallel sine and cosine processing paths). In this report, only real-value time signals will be treated, but in order to obtain maximum bandwidth efficiency from OFDM, the complex time signal may be preferred (possibly an analagous situation to QPSK vs. BPSK). Equation 2, for the complex time waveform, has all IFFT bins available as carriers except the DC bin.

Both IFFT size and assignment (selection) of carriers can be dynamic. The transmitter and receiver just have to use the same parameters. This is one of the advantages of OFDM. Its bandwidth usage (and bit rate) can be varied according to varying user requirements. A simple control message from a base station can change a mobile unit's IFFT size and carrier selection.

2.3 Modulation

Binary data from a memory device or from a digital processing stream is used as the modulating (baseband) signal. The following steps may be carried out in order to apply modulation to the carriers in OFDM:

- combine the binary data into symbols according to the number of bits/symbol selected
- convert the serial symbol stream into parallel segments according to the number of carriers, and form carrier symbol sequences
- apply differential coding to each carrier symbol sequence
- convert each symbol into a complex phase representation
- assign each carrier sequence to the appropriate IFFT bin, including the complex conjugates
- take the IFFT of the result

This is the same modulation technique described in Reference [3]. The Reference [2] matlab program carries out these steps and provides detailed commentary and examples for each one.

OFDM modulation is applied in the frequency domain. Figure 2 and Figure 3 give an example of modulated OFDM carriers for one symbol period, prior to IFFT. For this example, there are 4 carriers, the IFFT bin size is 64, and there is only 1 bit per symbol. The magnitude of each carrier is 1, but it could be scaled to any value. The phase for each carrier is either 0 or 180 degrees, according to the symbol being sent. The phase determines the value of the symbol (binary in this case, either a 1 or a 0). In the example, the first 3 bits (the first 3 carriers) are 0, and the 4^{th} bit (4^{th} carrier) is a 1.

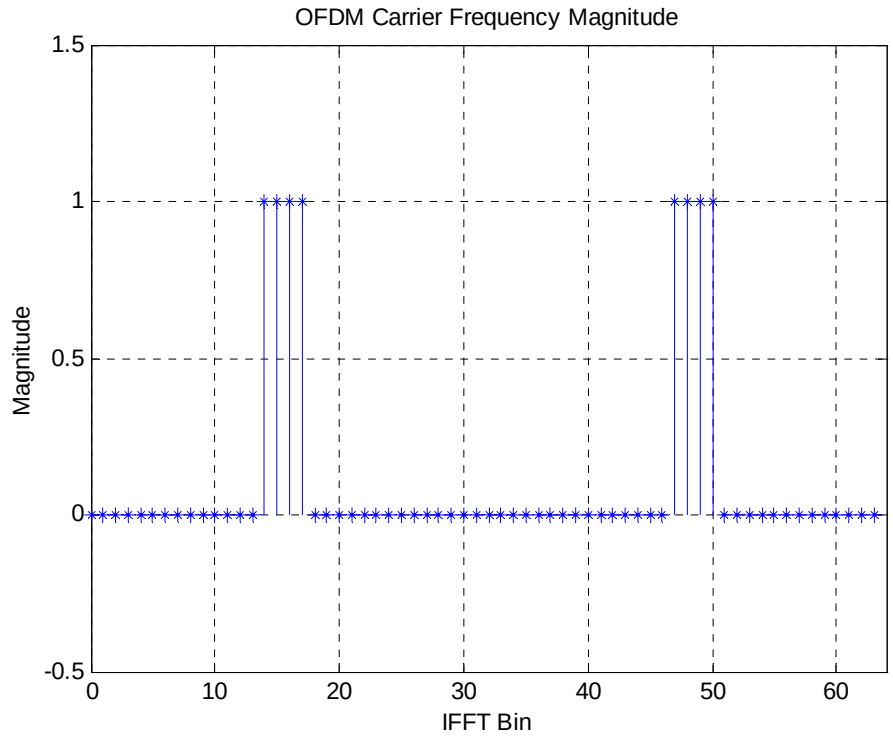

Figure 2: OFDM Carrier Magnitude prior to IFFT

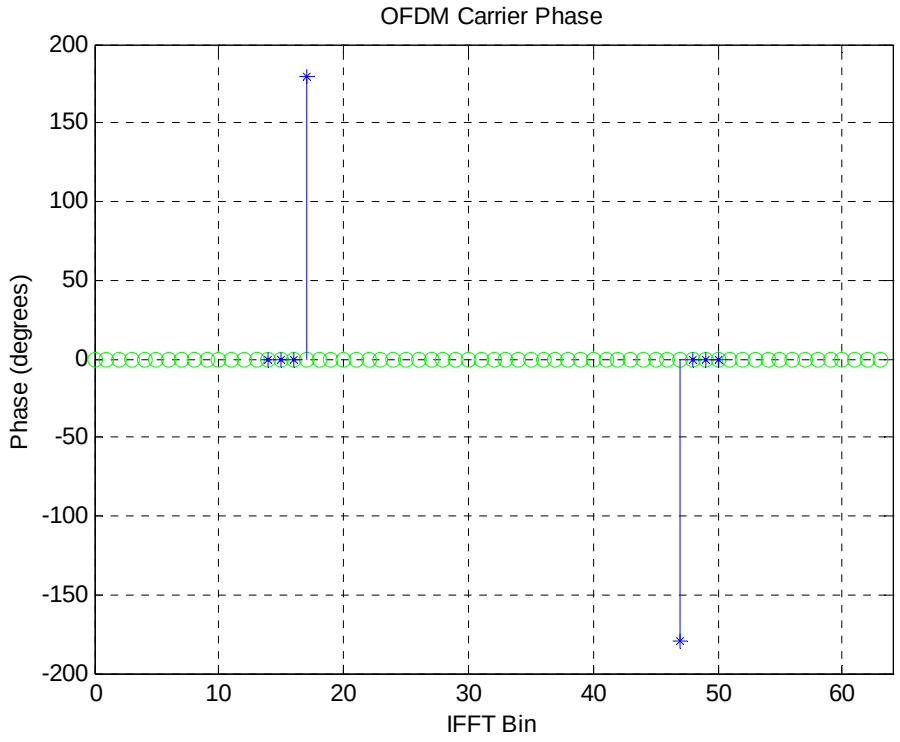

Figure 3: OFDM Carrier Phase prior to IFFT

Note that the modulated OFDM signal is nothing more than a group of delta (impulse) functions, each with a phase determined by the modulating symbol. In addition, note that the frequency separation between each delta is proportional to 1/N where N is the number of IFFT bins. The frequency domain representation of the OFDM is described in Equation 3.

Equation 3 : OFDM Frequency Domain Representation (one symbol period)

$$S(k) = e^{j\theta_m}\delta\left(k - m - \frac{N}{2}\right) + e^{-j\theta_m}\delta\left(k + m - \frac{N}{2}\right) \quad \text{single (real) OFDM modulated carrier}$$

k = frequency (0 to N - 1)

m = OFDM carrier frequency

N = IFFT bin size

$$S(k)_{ofdm} = \sum_{m=c_{first}}^{c_{last}} \left[e^{j\theta_m}\delta\left(k - m - \frac{N}{2}\right) + e^{-j\theta_m}\delta\left(k + m - \frac{N}{2}\right) \right] \quad \text{composite (real) OFDM modulated carriers}$$

c = OFDM carrier (first through last)

After the modulation is applied, an IFFT is performed to generate one symbol period in the time domain. The IFFT result is shown in Figure 4. It is clear that the OFDM signal has a varying amplitude. It is very important that the amplitude variations be kept intact as they define the content of the signal. If the amplitude is clipped or modified, then an FFT of the signal would no longer result in the original frequency characteristics, and the modulation may be lost.

This is one of the drawbacks of OFDM, the fact that it requires linear amplification. In addition, very large amplitude peaks may occur depending on how the sinusoids line up, so the peak-to-average power ratio is high. This means that the linear amplifier has to have a large dynamic range to avoid distorting the peaks. The result is a linear amplifier with a constant, high bias current resulting in very poor power efficiency. For a detailed treatment of the peak-to-average power ratio problem in OFDM, see Reference [4].

Figure 5 is provided to illustrate the time components of the OFDM signal. The IFFT transforms each complex conjugate pair of delta functions (each carrier) into a real-valued, pure sinusoid. Figure 5 shows the separate sinusoids that make up the composite OFDM waveform given in Figure 4. The one sinusoid with 180 phase shift is clearly visible as is the frequency difference between each of the 4 sinusoids. Note that this figure is 'zoomed' i.e. all 64 point of the IFFT are not shown. In addition, note that the waveform plots are not very smooth. This is because there are not many samples per cycle for any of the sinusoids.

The time domain representation of the OFDM signal is given in Equation 4.

Equation 4: OFDM Time Domain Representation (one symbol period)

$$s(n) = \sum_{m=c_{first}}^{c_{last}} \sum_{n=0}^{N-1} \cos\left(\frac{2\pi mn}{N} + \theta_m\right)$$

n = time sample

m = OFDM carrier

N = IFFT bin size

θ_m = phase modulation for OFDM carrier (m)

c_{first}, c_{last} = OFDM carriers (first and last)

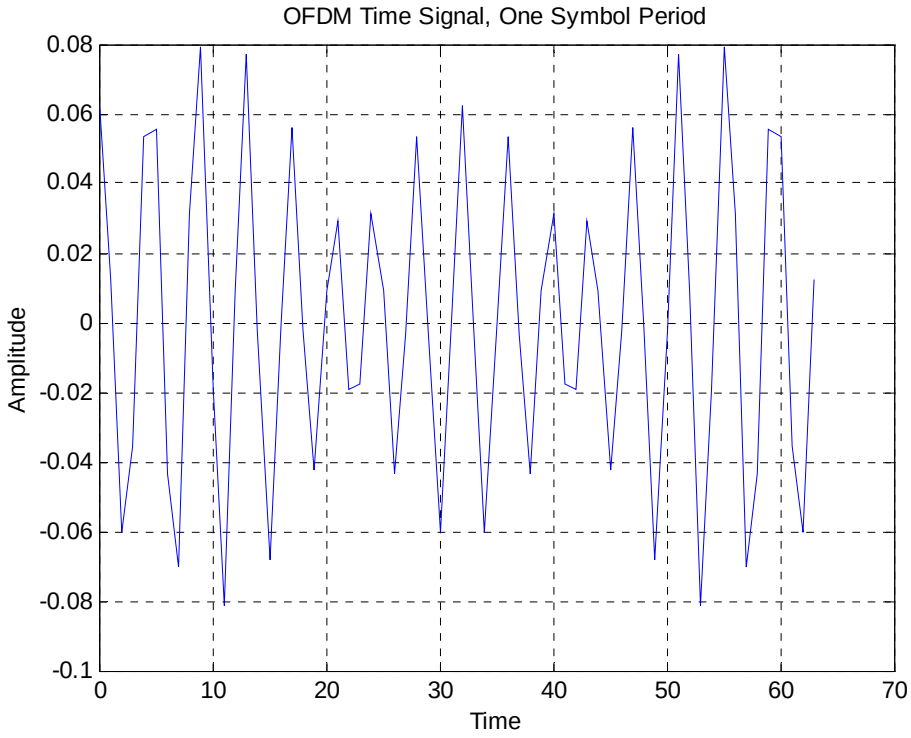

Figure 4: OFDM Signal, 1 Symbol Period

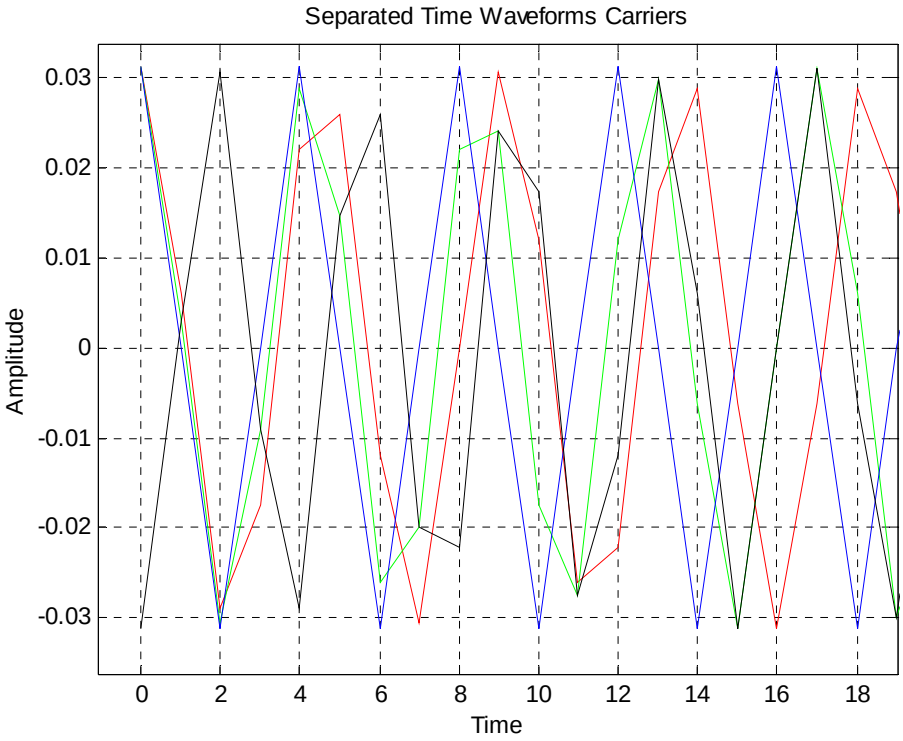

Figure 5: Separated Components of the OFDM Time Waveform

2.4 Transmission

The key to the uniqueness and desirability of OFDM is the relationship between the carrier frequencies and the symbol rate. Each carrier frequency is separated by a multiple of 1/NT (Hz). The symbol rate (R) for each carrier is 1/NT (symbols/sec).

The effect of the symbol rate on each OFDM carrier is to add a sin(x)/x shape to each carrier's spectrum. The nulls of the sin(x)/x (for each carrier) are at integer multiples of 1/NT. The peak (for each carrier) is at the carrier frequency k/NT. Therefore, each carrier frequency is located at the nulls for all the other carriers. This means that none of the carriers will interfere with each other during transmission, although their spectrums overlap. The ability to space carriers so closely together is very bandwidth efficient.

Figure 7 shows the spectrum for of an OFDM signal with the following characteristics:

- 1 bit / symbol
- 100 symbols / carrier (i.e. a sequence of 100 symbol periods)
- 4 carriers
- 64 IFFT bins
- spectrum averaged for every 20 symbols (100/20 = 5 averages)

Red diamonds mark all of the available carrier frequencies. Note that the nulls of the spectrums line up with the unused frequencies. The four active carriers each have peaks at carrier frequencies. It is clear that the active carriers have nulls in their spectrums at each of the unused frequencies (otherwise, the nulls would not exist). Although it cannot be seen in the figure, the active frequencies also have spectral nulls at the adjacent active frequencies.

Figure 6 shows the OFDM time waveform for the same signal. There are 100 symbol periods in the signal. Each symbol period is 64 samples long (100 x 64 = 6400 total samples). Each symbol period contains 4 carriers each of which carries 1 symbol. Each symbol carries 1 bit. Note that Figure 6 again illustrates the large dynamic range of the OFDM waveform envelope.

It is not currently practical to generate the OFDM signal directly at RF rates, so it must be upconverted for transmission. To remain in the discrete domain, the OFDM could be upsampled and added to a discrete carrier frequency. This carrier could be an intermediate frequency whose sample rate is handled by current technology. It could then be converted to analog and increased to the final transmit frequency using analog frequency conversion methods. Alternatively, the OFDM modulation could be immediately converted to analog and directly increased to the desired RF transmit frequency. Either way, the selected technique would have to involve some form of linear AM (possibly implemented with a mixer).

2.5 Reception and Demodulation

The received OFDM signal is downconverted (in frequency) and taken from analog to digital. Demodulation is done in the frequency domain (just as modulation was). The following steps may be taken to demodulate the OFDM:

- partition the input stream into vectors representing each symbol period
- take the FFT of each symbol period vector
- extract the carrier FFT bins and calculate the phase of each
- calculate the phase difference, from one symbol period to the next, for each carrier
- decode each phase into binary data
- sort the data into the appropriate order

The Reference [2] matlab program carries out these steps and provides detailed commentary and examples for each one. Figure 8 and Figure 9 show the magnitude and spectrum of the FFT for one received OFDM symbol period. For this example, there are 4 carriers, the IFFT bin size is 64, there is 1 bit per symbol, and the signal was sent through a channel with AWGN having an SNR of 8 dB. The figures show that, under these conditions, the modulated symbols are very easy to recover. Note in Figure 9 that the unused frequency bins contain widely varying phase values. These bins are not decoded, so it does not matter, but the result is of interest. Even if the noise is removed from the channel, these phase variations still occur. It must be a result of the IFFT/FFT operations generating very small complex values (very close to 0) for the unused carriers. The phases are a result of these values.

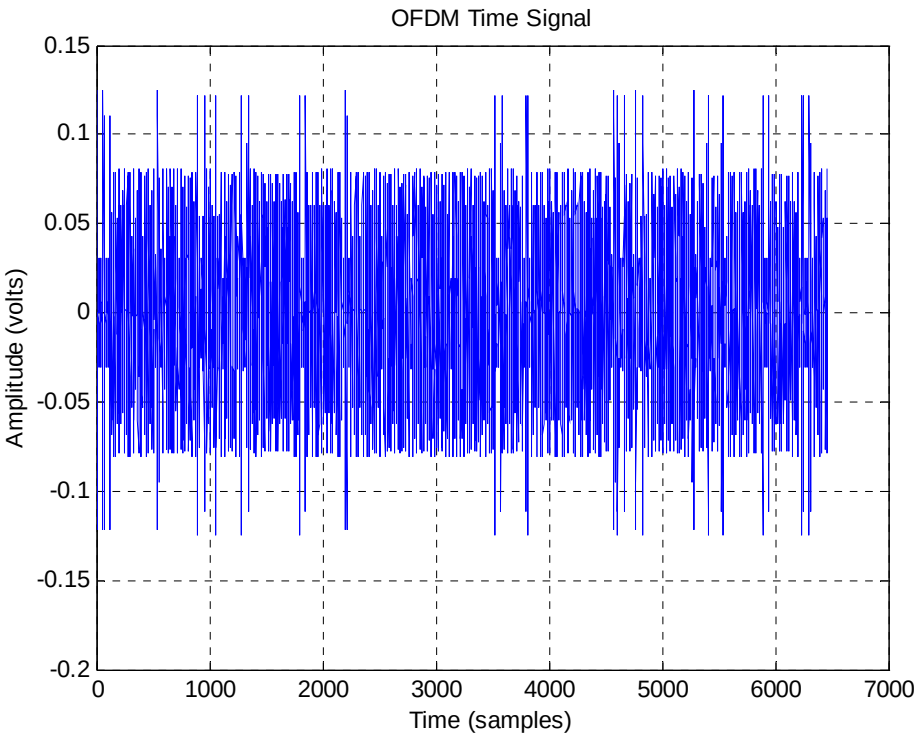

Figure 6: OFDM Time Waveform

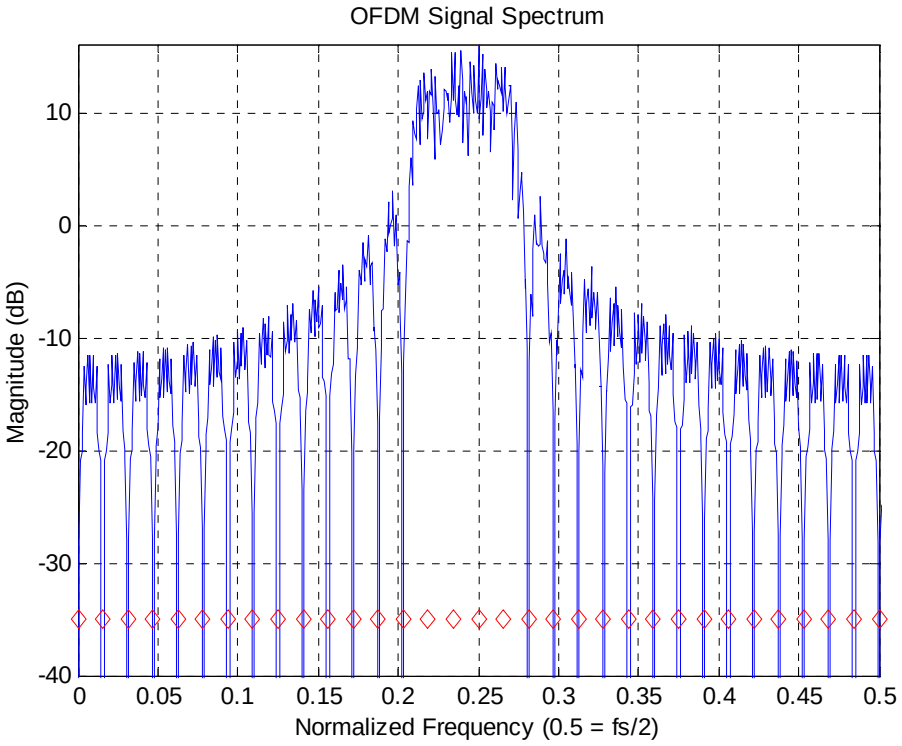

Figure 7: OFDM Spectrum

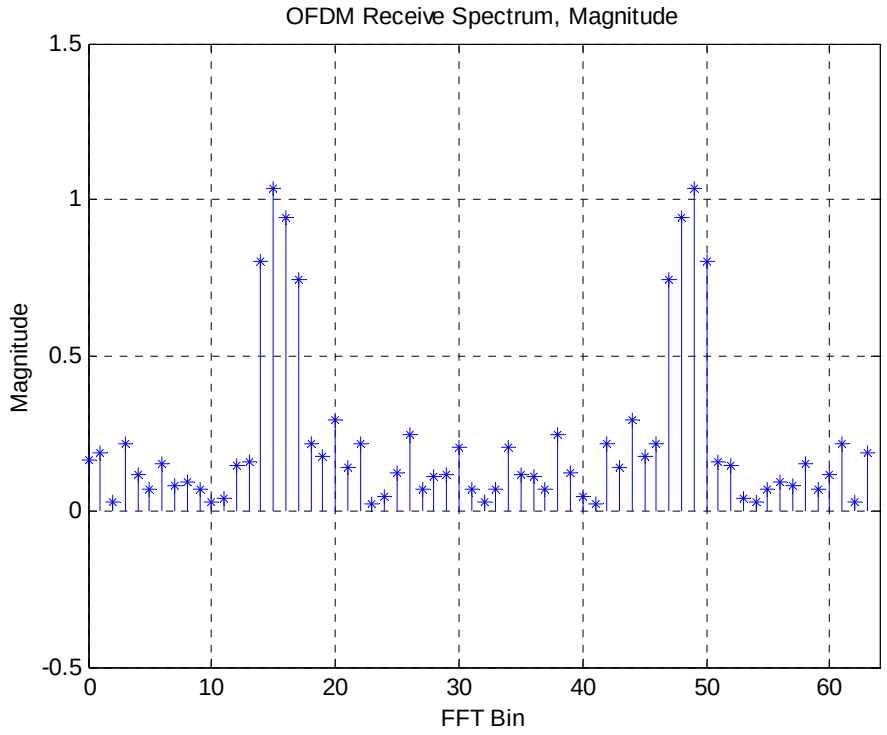

Figure 8: OFDM Carrier Magnitude following FFT

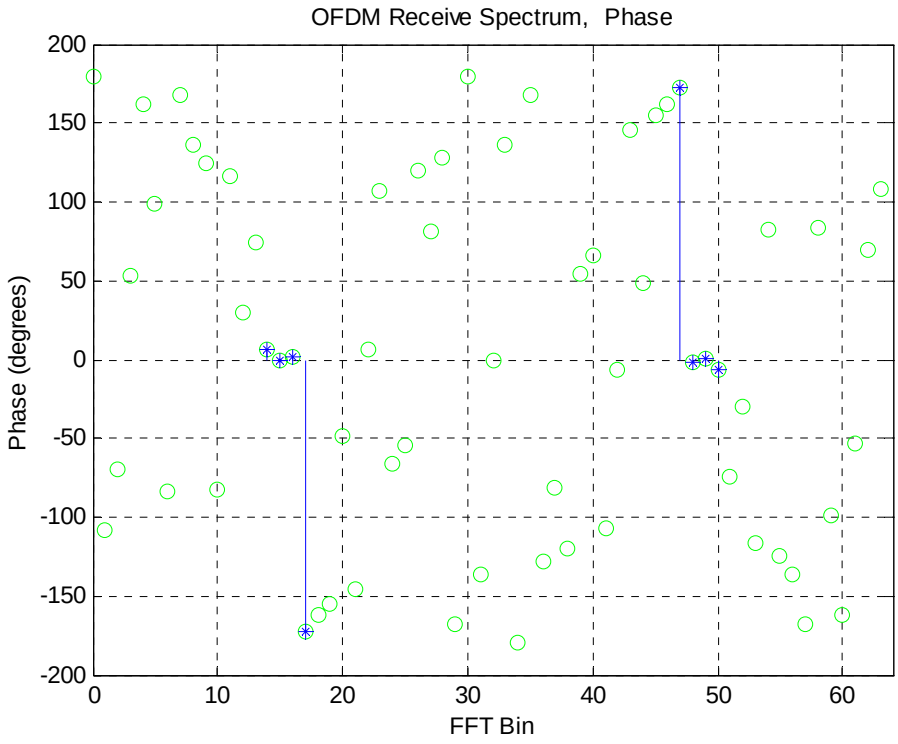

Figure 9: OFDM Carrier Phase following FFT

3 ANALYSIS

3.1 Guard Period

OFDM demodulation must be synchronized with the start and end of the transmitted symbol period. If it is not, then ISI will occur (since information will be decoded and combined for 2 adjacent symbol periods). ICI will also occur because orthogonality will be lost (integrals of the carrier products will no longer be zero over the integration period), Reference [5].

To help solve this problem, a guard interval is added to each OFDM symbol period. The first thought of how to do this might be to simply make the symbol period longer, so that the demodulator does not have to be so precise in picking the period beginning and end, and decoding is always done inside a single period. This would fix the ISI problem, but not the ICI problem. If a complete period is not integrated (via FFT), orthogonality will be lost.

In order to avoid ISI and ICI, the guard period must be formed by a cyclic extension of the symbol period. This is done by taking symbol period samples from the end of the period and appending them to the front of the period. The concept of being able to do this, and what it means, comes from the nature of the IFFT/FFT process. When the IFFT is taken for a symbol period (during OFDM modulation), the resulting time sample sequence is technically periodic. This is because the IFFT/FFT is an extension of the Fourier Transform which is an extension of the Fourier Series for periodic waveforms. All of these transforms operate on signals with either real or manufactured periodicity. For the IFFT/FFT, the period is the number of samples used. Reference [6] provides an excellent explanation of the Fourier Series and its extensions.

With the cyclic extension, the symbol period is longer, but it represents the exact same frequency spectrum. As long as the correct number of samples are taken for the decode, they may be taken anywhere within the extended symbol. Since a complete period is integrated, orthogonality is maintained. Therefore, both ISI and ICI are eliminated.

Note that some bandwidth efficiency is lost with the addition of the guard period (symbol period is increased and symbol rate is decreased).

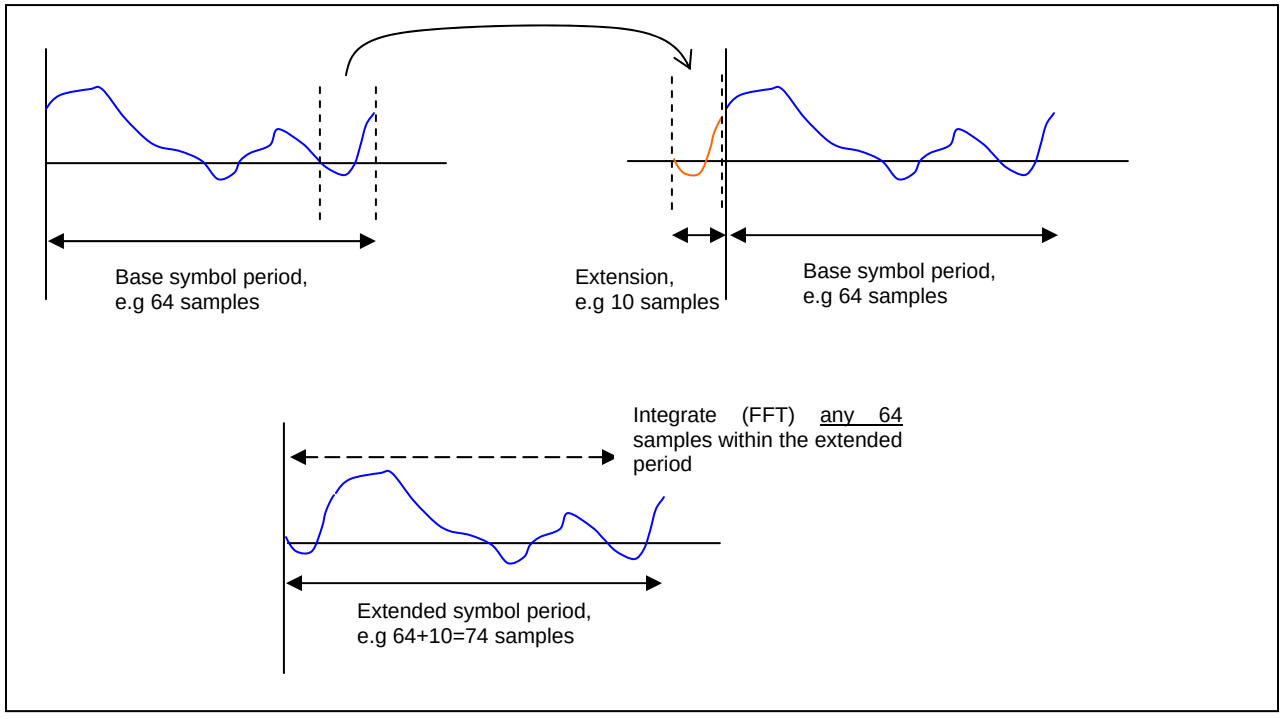

Figure 10: Guard Period via Cyclic Extension

3.2 Windowing

The OFDM signal is made up of a series of IFFTs that are concatenated to each other. At each symbol period boundary, there is a signal discontinuity due to the differences between the end of one period and the start of the next. These discontinuities can cause high frequency spectral noise to be generated (because they look like very fast transitions of the time waveform). To avoid this, a window function (Hamming, Hanning, Blackman, …) may be applied to each symbol period. The window function would attenuate the time waveform at the start and the end of each period, so that the discontinuities are smaller, and the high frequency noise is reduced. However, this attenuation distorts the signal and some of the desired frequency content is lost.

Reference [3] recommends using a window function, and one is included in the Reference [2] matlab program, but it is commented out. It is not clear whether the window function helps more than it hurts. If it does not add sufficient benefit, it should not be included as it is an extra processing step that would increase the load on the OFDM generation process (an extra multiply operation on each symbol period vector).

3.3 Multipath Characteristics

OFDM avoids frequency selective fading and ISI by providing relatively long symbol periods for a given data rate. This is illustrated in Figure 11. For a given transmission channel and a given source data rate, OFDM can provide better multipath characteristics than a single carrier.

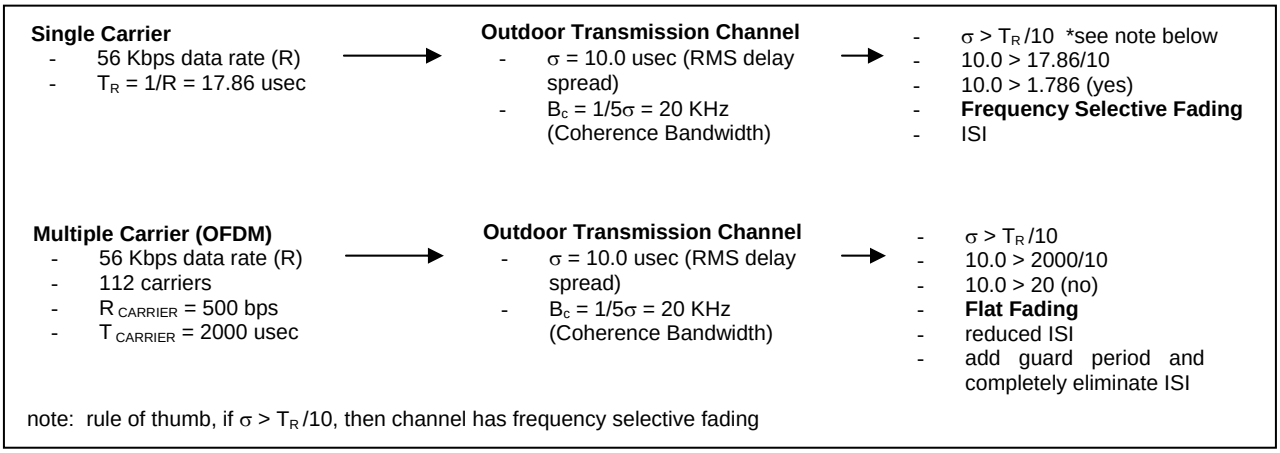

Figure 11: OFDM vs. Single Carrier, Multipath Characteristic Comparison

However, since the OFDM carriers are spread over a frequency range, there still may be some frequency selective attenuation on a time-varying basis. A deep fade on a particular frequency may cause the loss of data on that frequency for a given time, but the use of Forward Error Coding can fix it. If a single carrier experienced a deep fade, too many consecutive symbols may be lost and correction coding may be ineffective. For more detail on OFDM error coding techniques, see Reference [5].

3.4 Bandwidth

A comparison of RF transmit bandwidth between OFDM and a single carrier is shown in Figure 12 (using the same example parameters as in Figure 11). The calculations show that OFDM is more bandwidth efficient than a single carrier. Note that another efficient aspect of OFDM is that a single transmitter's bandwidth can be increased incrementally by addition of more adjacent carriers. In addition, no bandwidth buffers are needed between transmit bandwidths of separate transmitters as long as orthogonality can be maintained between all the carriers.

Single Carrier

- 56 Kbps data rate (R)
- T_R = 1/R = 17.86 usec
- Raised Cosine Filter pulse shaping, α = 0.3
- BW_{RF} = R(1+α) = 56 Kbps (1+0.3)
- BW_{RF} = 72.8 KHz

Multiple Carrier (OFDM)

- 56 Kbps data rate (R)
- 112 carriers
- $R_{CARRIER}$ = 500 bps
- $T_{CARRIER}$ = 2000 usec
- N = IFFT bin count
- T = base period
- NT = symbol period = $T_{CARRIER}$ = 2000 usec
- n = number of carriers = 112
- BW_{RF} = (n+1)/NT
- BW_{RF} = (112+1)/2000 usec
- BW_{RF} = 56.5 KHz

OFDM is 29% more bandwidth efficient for this example (72.8/56.5 = 1.2885)

Figure 12: OFDM Bandwidth Efficiency

3.5 Physical Implementation

Since OFDM is carried out in the digital domain, there are many ways it can be implemented. Some options are provided in the following list. Each of these options should be viable given current technology:

1. ASIC (Application Specific Integrated Circuit)
   - ASICs are the fastest, smallest, and lowest power way to implement OFDM
   - Cannot change the ASIC after it is built without designing a new chip
2. General-purpose Microprocessor or MicroController
   - PowerPC 7400 or other processor capable of fast vector operations
   - Highly programmable
   - Needs memory and other peripheral chips
   - Uses the most power and space, and would be the slowest
3. Field-Programmable Gate Array (FPGA)
   - An FPGA combines the speed, power, and density attributes of an ASIC with the programmability of a general purpose processor.
   - An FPGA could be reprogrammed for new functions by a base station to meet future (currently unknown requirements).
   - This should be the best choice

3.6 Applications

OFDM applications include the following [7,8]:

1. Digital Audio Broadcasting (DAB), wireless CD-quality sound transmission
2. Digital Video Broadcasting (DVB), specifically, Digital Terrestrial Television Broadcasting (DTTB)
3. Wireless LAN (IEEE 802.11a)
4. ADSL (Asymmetric Digital Subscriber Line), also called DMT (Digital Multi-Tone)

4 REFERENCES

[1] Wide-band Orthogonal Frequency Multiplexing (W-OFDM), White Paper by Wireless Data Communications Inc., www.wi-lan.com.

[2] OFDM Simulation (matlab), Erich Cosby, 2001 (based on Reference [3]), this matlab file is included as Section 5 of this report.

[3] The suitability of OFDM as a modulation technique for wireless telecommunications, with a CDMA comparison, Eric Lawrey, 1997, www.eng.jcu.edu.au/eric/thesis/Thesis.htm.

[4] Orthogonal Frequency Division Multiplexing (OFDM) – Applications for Wireless Communications with Coding, unknown author, www.comlab.hut.fi/opetus/311/ofdm_mod.pdf.

[5] The how and why of COFDM, J. H. Stott, 1998, www.bbc.co.uk/rd/pubs/papers/pdffiles/ptrev_278-stott.pdf.

[6] Who Is Fourier?, The Transnational College of Lex, 1995, Language Research Foundation (available at amazon.com).

[7] Orthogonal Frequency Division Multiplexing (OFDM) Explained, Magis Networks, Inc., February 8, 2001, www.magisnetworks.com.

[8] OFDM Tutorial, Wave Report, www.wave-report.com/tutorials/OFDM.htm.

5 MATLAB

The matlab file allows the user to specify OFDM parameters and analyze the resulting waveforms. There are embedded plot sequences that can be 'uncommented' to analyze the OFDM processing path at different points in the sequence.

```
%--------1---------2---------3---------4---------5---------6---------7---------8
% OFDM Simulation
%
%    - ECE 5664 Project
%    - Erich Cosby
%    - December 4, 2001
%    - Virginia Tech, Nothern Va. Center
%
% Based on the matlab work of Eric Lawrey in his 1997 BSEE thesis,
% "The suitability of OFDM as a modulation technique for wireless
%  telecommunications, with a CDMA comparison",
% refer to www.eng.jcu.edu.au/eric/thesis/Thesis.htm.
%
%
clear all;
close all;
%
% Basic OFDM system parameters
%  - choice of defaults or user selections
%
fprintf ('OFDM Analysis Program\n\n');
defaults = input('To use default parameters, input "1", otherwise input "0":  ');
%
if defaults == 1
   IFFT_bin_length = 1024;     % IFFT bin count for Tx, FFT bin count for Rx
   carrier_count = 200;        % number of carriers
   bits_per_symbol = 2;        % bits per symbol
   symbols_per_carrier = 50;   % symbols per carrier
   SNR = 10;                   % channel signal to noise ratio (dB)
else
   IFFT_bin_length = input('IFFT bin length = ');
   carrier_count = input('carrier count = ');
   bits_per_symbol = input('bits per symbol = ');
   symbols_per_carrier = input('symbols per carrier =');
   SNR = input('SNR = ');
end
%
% Derived parameters
%
baseband_out_length =  carrier_count * symbols_per_carrier * bits_per_symbol;
carriers = (1:carrier_count) + (floor(IFFT_bin_length/4) - floor(carrier_count/2));
conjugate_carriers = IFFT_bin_length - carriers + 2;
%
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
% TRANSMIT >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%
%
% Generate a random binary output signal:
%   - a row of uniform random numbers (between 0 and 1), rounded to 0 or 1
%   - this will be the baseband signal which is to be transmitted.
%
baseband_out = round(rand(1,baseband_out_length));
```

```
%
% Convert to 'modulo N' integers where N = 2^bits_per_symbol
%   - this defines how many states each symbol can represent
%   - first, make a matrix with each column representing consecutive bits
%     from the input stream and the number of bits in a column equal to the
%     number of bits per symbol
%   - then, for each column, multiply each row value by the power of 2 that
%     it represents and add all the rows
%   - for example:  input 0 1 1 0 0 0 1 1 1 0
%                   bits_per_symbol = 2
%                   convert_matrix = 0 1 0 1 1
%                                    1 0 0 1 0
%
%                   modulo_baseband = 1 2 0 3 2
%
convert_matrix = reshape(baseband_out, bits_per_symbol, length(baseband_out)/bits_per_symbol);
for k = 1:(length(baseband_out)/bits_per_symbol)
   modulo_baseband(k) = 0;
   for i = 1:bits_per_symbol
      modulo_baseband(k) = modulo_baseband(k) + convert_matrix(i,k)*2^(bits_per_symbol-i);
   end
end
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
% Serial to Parallel Conversion
%   - convert the serial modulo N stream into a matrix where each column
%     represents a carrier and each row represents a symbol
%   - for example:
%
%     serial input stream = a b c d e f g h i j k l m n o p
%
%     parallel carrier distribution =
%                       C1/s1=a  C2/s1=b  C3/s1=c C4/s1=d
%                       C1/s2=e  C2/s2=f  C3/s2=g C4/s2=h
%                       C1/s3=i  C2/s3=j  C3/s3=k C4/s3=l
%                         .       .       .      .
%                         .       .       .      .
%
carrier_matrix = reshape(modulo_baseband, carrier_count, symbols_per_carrier)';
%
% Apply differential coding to each carrier string
%   - append an arbitrary start symbol (let it be 0, that works for all
%     values of bits_per_symbol) (note that this is done using a vertical
%     concatenation [x;y] of a row of zeros with the carrier matrix, sweet!)
%   - perform modulo N addition between symbol(n) and symbol(n-1) to get the
%     coded value of symbol(n)
%   - for example:
%                   bits_per_symbol = 2 (modulo 4)
%                   symbol stream =  3 2 1 0 2 3
%                   start symbol = 0
%
%                   coded symbols = 0 + 3 = 3
%                                   3 + 2 = 11 = 1
%                                   1 + 1 = 2
%                                   2 + 0 = 2
%                                   2 + 2 = 10 = 0
%                                   0 + 3 = 3
%
%                   coded stream =  0 3 1 2 2 0 3
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
```

```
carrier_matrix = [zeros(1,carrier_count);carrier_matrix];
for i = 2:(symbols_per_carrier + 1)
   carrier_matrix(i,:) = rem(carrier_matrix(i,:)+carrier_matrix(i-1,:),2^bits_per_symbol);
end
%
% Convert the differential coding into a phase
%   - each phase represents a different state of the symbol
%   - for example:
%               bits_per_symbol = 2 (modulo 4)
%               symbols = 0 3 2 1
%               phases =
%                       0 * 2pi/4 = 0 (0 degrees)
%                       3 * 2pi/4 = 3pi/2 (270 degrees)
%                       2 * 2pi/4 = pi (180 degrees)
%                       1 * 2pi/4 = pi/2 (90 degrees)
%
carrier_matrix = carrier_matrix * ((2*pi)/(2^bits_per_symbol));
%
% Convert the phase to a complex number
%   - each symbol is given a magnitude of 1 to go along with its phase
%     (via the ones(r,c) function)
%   - it is then converted from polar to cartesian (complex) form
%   - the result is 2 matrices, X with the real values and Y with the imaginary
%   - each X column has all the real values for a carrier, and each Y column
%     has the imaginary values for a carrier
%   - a single complex matrix is then generated taking X for real and
%     Y for imaginary
%
[X,Y] = pol2cart(carrier_matrix, ones(size(carrier_matrix,1),size(carrier_matrix,2)));
complex_carrier_matrix = complex(X,Y);
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
% Assign each carrier to its IFFT bin
%   - each row of complex_carrier_matrix represents one symbol period, with
%     a symbol for each carrier
%   - a matrix is generated to represent all IFFT bins (columns) and all
%     symbols (rows)
%   - the phase modulation for each carrier is then assigned to the
%     appropriate bin
%   - the conjugate of the phase modulation is then assigned to the
%     appropriate bin
%      - the phase modulation bins and their conjugates are symmetric about
%        the Nyquist frequency in the IFFT bins
%      - since the first bin is DC, the Nyquist Frequency is located
%        at (number of bins/2) + 1
%      - symmetric conjugates are generated so that when the signal is
%        transformed to the time domain, the time signal will be real-valued
%   - example
%      - 1024 IFFT bins
%      - bin 513 is the center (symmetry point)
%      - bin 1 is DC
%      - bin 514 is the complex conjugate of bin 512
%      - bin 515 is the complex conjugate of bin 511
%      - ....
%      - bin 1024 is the complex conjugate of bin 2 (if all bins
%        were used as carriers)
%      - So, bins 2-512 map to bins 1024-514
%
IFFT_modulation = zeros(symbols_per_carrier + 1, IFFT_bin_length);
IFFT_modulation(:,carriers) = complex_carrier_matrix;
IFFT_modulation(:,conjugate_carriers) = conj(complex_carrier_matrix);
%
```

```
% PLOT BASIC FREQUENCY DOMAIN REPRESENTATION
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
%figure (1)
%stem(0:IFFT_bin_length-1, abs(IFFT_modulation(2,1:IFFT_bin_length)),'b*-')
%grid on
%axis ([0 IFFT_bin_length -0.5 1.5])
%ylabel('Magnitude')
%xlabel('IFFT Bin')
%title('OFDM Carrier Frequency Magnitude')
%figure (2)
%plot(0:IFFT_bin_length-1, (180/pi)*angle(IFFT_modulation(2,1:IFFT_bin_length)), 'go')
%hold on
%stem(carriers-1, (180/pi)*angle(IFFT_modulation(2,carriers)),'b*-')
%stem(conjugate_carriers-1, (180/pi)*angle(IFFT_modulation(2,conjugate_carriers)),'b*-')
%axis ([0 IFFT_bin_length -200 +200])
%grid on
%ylabel('Phase (degrees)')
%xlabel('IFFT Bin')
%title('OFDM Carrier Phase')
% END OF PLOTTING
%
% Transform each period's spectrum (represented by a row of carriers) to the
% time domain via IFFT
%
time_wave_matrix = ifft(IFFT_modulation');
time_wave_matrix = time_wave_matrix';
%
% PLOT OFDM SIGNAL FOR ONE SYMBOL PERIOD
%   - first plot is direct IFFT of first OFDM symbol (index 2)
%     - index 1 is the 'start' symbol due to differential implementation
%   - second plot strips out each carrier and plots them on the
%     same graph
%     - only works for carrier count <= 16 due to colors variable (more
%       than 16 would really be legible anyway
%
%figure (3)
%plot(0:IFFT_bin_length-1,time_wave_matrix(2,:))
%grid on
%ylabel('Amplitude')
%xlabel('Time')
%title('OFDM Time Signal, One Symbol Period')
%
%colors = ['r' 'g' 'b' 'k' 'r' 'g' 'b' 'k' 'r' 'g' 'b' 'k' 'r' 'g' 'b' 'k'];
%for f = 1:carrier_count
%    temp_bins(1:IFFT_bin_length)=0+0j;
%    temp_bins(carriers(f))=IFFT_modulation(2,carriers(f));
%    temp_bins(conjugate_carriers(f))=IFFT_modulation(2,conjugate_carriers(f));
%    temp_time = ifft(temp_bins');
%    figure(4)
%    plot(0:IFFT_bin_length-1, temp_time, colors(f))
%   hold on
%end
%grid on
%ylabel('Amplitude')
%xlabel('Time')
%title('Separated Time Waveforms Carriers')
%
% END OF PLOTTING
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
```

```
% Apply a Window Function to each time waveform
%   - NOTE THAT WINDOWING IS CURRENTLY COMMENTED OUT, i.e. NO WINDOWING
%   - each time waveform (row of time_wave_matrix) represents one symbol
%     period for all carriers
%   - the IFFT result has discontinuities at each end
%   - when the time waveforms are serialized (concatenated), the discontinuites
%     will introduce unwanted frequency components
%   - the window function deemphasizes the signal at the end
%     points (at the discontinuites)
%      - this reduces the effects of the discontinuities
%      - it also distorts the desired frequency response (undesired side effect)
%   - between Blackman, Hanning, and Hamming:  Hamming introduces less distortion
%   - note that the transpose of the Hamming function is
%     used (because a row vector is needed)
%
% Since all imaginary values of time_wave_matrix are practically equal to zero,
% only the real part is retained for windowing.
%
for i = 1:symbols_per_carrier + 1
    %windowed_time_wave_matrix(i,:) = real(time_wave_matrix(i,:)) .* hamming(IFFT_bin_length)';
    windowed_time_wave_matrix(i,:) = real(time_wave_matrix(i,:));
end
%
% Serialize the modulating waveform
%   - sequentially take each row of windowed_time_wave_matrix and construct a row vector
%   - the row vector will be the modulating signal
%   - note that windowed_time_wave_matrix is transposed, this is to account for the way the
%     Matlab 'reshape' function works (reshape takes the columns of the target matrix and
%     appends them sequentially)
%
ofdm_modulation = reshape(windowed_time_wave_matrix', 1, IFFT_bin_length*(symbols_per_carrier+1));
%
% PLOT OFDM SIGNAL (time)
%
%temp_time = IFFT_bin_length*(symbols_per_carrier+1);
%figure (5)
%plot(0:temp_time-1,ofdm_modulation)
%grid on
%ylabel('Amplitude (volts)')
%xlabel('Time (samples)')
%title('OFDM Time Signal')
%
% PLOT OFDM SIGNAL (spectrum)
%symbols_per_average = ceil(symbols_per_carrier/5);
%avg_temp_time = IFFT_bin_length*symbols_per_average;
%averages = floor(temp_time/avg_temp_time);
%average_fft(1:avg_temp_time) = 0;
%for a = 0:(averages-1)
%    subset_ofdm = ofdm_modulation(((a*avg_temp_time)+1):((a+1)*avg_temp_time));
%    subset_ofdm_f = abs(fft(subset_ofdm));
%    average_fft = average_fft + (subset_ofdm_f/averages);
%end
%average_fft_log = 20*log10(average_fft);
%figure (6)
%plot((0:(avg_temp_time-1))/avg_temp_time, average_fft_log)
%hold on
%plot(0:1/IFFT_bin_length:1, -35, 'rd')
%grid on
%axis([0 0.5 -40 max(average_fft_log)])
%ylabel('Magnitude (dB)')
%xlabel('Normalized Frequency (0.5 = fs/2)')
%title('OFDM Signal Spectrum')
%
```

```
% ENDPLOT
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
% Upconversion to RF
%
% For this model, the baseband will be inserted directly into the channel
% without conversion to RF frequencies.
%
Tx_data = ofdm_modulation;
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
% CHANNEL ============================================================================
%
% The channel model is Gaussian (AWGN) only
%   - Rayleigh fading would be a useful addition
%
Tx_signal_power = var(Tx_data);
%
linear_SNR = 10^(SNR/10);
noise_sigma = Tx_signal_power/linear_SNR;
noise_scale_factor = sqrt(noise_sigma);
%
noise = randn(1, length(Tx_data))*noise_scale_factor;
Rx_Data = Tx_data + noise;
%
%
% RECEIVE <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%
%
% Convert the serial input data stream to parallel (according to symbol length
% and number of symbols)
%   - each column is a symbol period
%   - the length of each symbol (samples per symbol) is the length of the
%     IFFT that was used to generate it
%
Rx_Data_matrix = reshape(Rx_Data, IFFT_bin_length, symbols_per_carrier + 1);
%
% Transform each symbol from time to frequency domain
%   - take the fft of each column
%
Rx_spectrum = fft(Rx_Data_matrix);%
% PLOT BASIC FREQUENCY DOMAIN REPRESENTATION
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
%figure (7)
%stem(0:IFFT_bin_length-1, abs(Rx_spectrum(1:IFFT_bin_length,2)),'b*-')
%grid on
%axis ([0 IFFT_bin_length -0.5 1.5])
%ylabel('Magnitude')
%xlabel('FFT Bin')
%title('OFDM Receive Spectrum, Magnitude')
%figure (8)
%plot(0:IFFT_bin_length-1, (180/pi)*angle(Rx_spectrum(1:IFFT_bin_length,2)), 'go')
%hold on
%stem(carriers-1, (180/pi)*angle(Rx_spectrum(carriers,2)),'b*-')
%stem(conjugate_carriers-1, (180/pi)*angle(Rx_spectrum(conjugate_carriers,2)),'b*-')
%axis ([0 IFFT_bin_length -200 +200])
%grid on
%ylabel('Phase (degrees)')
%xlabel('FFT Bin')
```

```
%title('OFDM Receive Spectrum,  Phase')
%
% END OF PLOTTING


%--------1---------2---------3---------4---------5---------6---------7---------8
%
% Extract the carrier FFT bins
%   - only keep the fft bins that are used as carriers
%   - take the transpose of the result so that each column will represent
%     a carrier
%     - this is in preparation for using the diff( ) function later to decode
%       differential encoding
%     - format following this operation is:
%
%             C1-s1  C2-s1  C3-s1 ...
%             C1-s2  C2-s2  C3-s2 ...
%             C1-s3  C2-s3  C3-s3 ...
%                .     .     .
%                .     .     .
%
%   - IMPORTANT MATLAB NOTE CONCERNING TRANSPOSING AND CONJUGATION
%     - it appears that each time a matrix is transposed, the conjugate of
%       each value is taken
%     - if an even number of transposes are done, then it is transparent
%     - obviously, this does not affect real numbers
%
Rx_carriers = Rx_spectrum(carriers,:)';
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
% PLOT EACH RECEIVED SYMBOL
%
figure (9)
Rx_phase_P = angle(Rx_carriers);
Rx_mag_P = abs(Rx_carriers);
polar(Rx_phase_P, Rx_mag_P,'bd');
%
% END PLOT
%
% Find the phase (angle) of each FFT bin (each carrier)
%   - convert from radians to degrees
%   - normalize phase to be between 0 and 359 degrees
%
Rx_phase = angle(Rx_carriers)*(180/pi);
phase_negative = find(Rx_phase < 0);
Rx_phase(phase_negative) = rem(Rx_phase(phase_negative)+360,360);
%
% Extract phase differences (from the differential encoding)
%   - the matlab diff( ) function is perfect for this operation
%   - again, normalize the result to be between 0 and 359 degrees
%
Rx_decoded_phase = diff(Rx_phase);
phase_negative = find(Rx_decoded_phase < 0);
Rx_decoded_phase(phase_negative) = rem(Rx_decoded_phase(phase_negative)+360,360);
%
%--------1---------2---------3---------4---------5---------6---------7---------8
%
% Convert phase to symbol
%   - calculate the base phase which is the phase difference between each
%     consecutive symbol
%   - for example, if there are 2 bits per symbol, base phase is 90 and the
%     symbols are represented by 0, 90, 180, and 270 degrees
```

```
%   - calculate the maximum deviation from the base phase that will still be
%     decoded as base phase
%   - for example, if base phase is 90, then delta phase is 45, and anything
%     within 45 degrees of base phase is accepted as base phase
%      - continuing the above example, a symbol represented by 180 will be
%        decoded as 180 as long as it is within the range 180+45 and 180-45
%   - generate a symbol matrix where the results of the phase decode
%     will be placed
%      - note that since the matrix is created as a zero matrix, then the zero
%        values do not have to be decoded
%      - zero is therefore the default value after decoding, if a value is not
%        decoded as anything else, then it is zero
%      - this actually save alot of trouble since the zero phase spans the
%        lowest and highest phase values and therefore requires special
%        processing
%      - it is also efficient in that it eliminates a pass through the loop
%
base_phase = 360/2^bits_per_symbol;
delta_phase = base_phase/2;
Rx_decoded_symbols = zeros(size(Rx_decoded_phase,1),size(Rx_decoded_phase,2));
%
for i = 1:(2^bits_per_symbol - 1)
   center_phase = base_phase*i;
   plus_delta = center_phase+delta_phase;
   minus_delta = center_phase-delta_phase;
   decoded = find((Rx_decoded_phase <= plus_delta) & (Rx_decoded_phase > minus_delta));
   Rx_decoded_symbols(decoded)=i;
end
%
% Convert the matrix into a serial symbol stream
%
Rx_serial_symbols = reshape(Rx_decoded_symbols',1,size(Rx_decoded_symbols,1)*size(Rx_decoded_symbols,2));
%
% Convert the symbols to binary
%
for i = bits_per_symbol: -1: 1
   if i ~= 1
      Rx_binary_matrix(i,:) = rem(Rx_serial_symbols,2);
      Rx_serial_symbols = floor(Rx_serial_symbols/2);
   else
      Rx_binary_matrix(i,:) = Rx_serial_symbols;
   end
end
baseband_in = reshape(Rx_binary_matrix,1,size(Rx_binary_matrix,1)*size(Rx_binary_matrix,2));
%
% Find bit errors
%
bit_errors = find(baseband_in ~= baseband_out);
bit_error_count = size(bit_errors,2);
%--------1---------2---------3---------4---------5---------6---------7---------8
```