Project #2

ECE-S 631:
Fundamentals of Deterministic DSP

Prof. John Walsh

Project Report

Zexi Liu

Nov 26th, 2009

Electrical and Computer Engineering
Drexel University

Project II: Multicarrier Equalization

ECES 631, Prof. John MacLaren Walsh, Ph. D.

1. Purpose

In this project you will encounter some of the practical utility of the theory that you have learned as part of ECES 631. In particular, you appreciate a particular practical use of the discrete Fourier transform (DFT) along with its efficient computation using the fast Fourier transform (FFT).You will encounter the differences between cyclic and linear convolution, and implement frequency domain equalization. Finally, you will hone your MATLAB signal processing skills to implement a simple multicarrier receiver.

2. Transmitter

The transmitter portion of a multicarrier communications system is pictured in Figure 1. The message that we wish to transmit is a finite duration signal

$$\mathbf{s} := [s[1], s[2], ..., s[MN]]$$

The serial to parallel convertor breaks this message up into M blocks of length N symbols (N is a power of 2) which we label as s_k

$$s_k := [s[Nk+1], s[Nk+2], ..., s[Nk+N]],\ k \in \{0, 1, ..., M-1\}$$

The receiver then computes the length N inverse DFT of the blocks s_k to get the signal x_k. Next, in the addition of the cyclic prefix, the last P symbols of each block x_k are appended to the beginning of the block to make blocks of length N + P, which we label as y_k. Finally, y is passed through a parallel to serial convertor to get a signal t.

$$\mathbf{t} := \left[\mathbf{y}_0^T, \mathbf{y}_1^T, \ldots, \mathbf{y}_{M-1}^T\right]^T$$

where T denotes the transpose operation which converts a row vector into a column vector. We will denote the nth element of the transmitted signal vector t as t[n].

1. The operation between the vectors s and t which the transmitter performs may be written as a matrix multiplication

   $$\mathbf{t} = \mathbf{Ms}$$

   Write the M as the multiplication of matrices **U** and **G** which describe the operation of the inverse DFT and the cyclic prefix addition operations, respectively. You may use the diag operation in

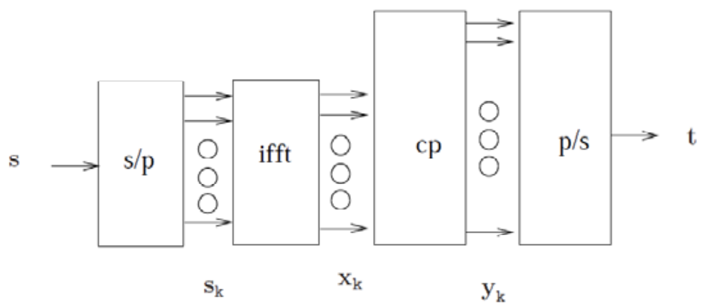


Figure 1: Transmitter portion of a simple multicarrier communications system.

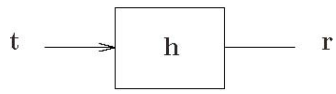


Figure 2: A simple model for a multipath communications channel.

your notation, which forms a block diagonal matrix whose block diagonal elements are the matrix arguments of diag. You may also use $\mathbf{I}_N$ to denote the identity matrix of length N and $\mathbf{0}_{N\times M}$ to denote a matrix of dimension N rows by M columns whose elements are all zero. You may also use a matrix $\mathbf{F}_N$ which is the N point DFT matrix, as long as you specify its element at any row i and any column j (with i, j $\in \{1, \dots, N\}$).

$$x_k = \begin{bmatrix} x_k[0] \\ \vdots \\ x_k[N-1] \end{bmatrix} = IDFT(s_k) = \frac{1}{N}\begin{bmatrix} \omega_N^{-0} & \cdots & \omega_N^{-0\cdot(N-1)} \\ \vdots & \ddots & \vdots \\ \omega_N^{-(N-1)\cdot 0} & \cdots & \omega_N^{-(N-1)\cdot(N-1)} \end{bmatrix} \cdot \begin{bmatrix} s_k[0] \\ \vdots \\ s_k[N-1] \end{bmatrix}$$

$$x_k = F'_{N\times N} \cdot s_k$$

$$F'_{N\times N} = \frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \cdots & \omega^{-(N-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} & \cdots & \omega^{-2(N-1)} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} & \cdots & \omega^{-3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(N-1)} & \omega^{-2(N-1)} & \omega^{-3(N-1)} & \cdots & \omega^{-(N-1)(N-1)} \end{bmatrix}_{N\times N}$$

$$F'_{ij} = \frac{1}{N}\omega_N^{-(i-1)(j-1)}, i \in \{1,2,3,\dots,N\}, j \in \{1,2,3,\dots,N\}$$

$$y_k = \begin{bmatrix} x_k[N-P] \\ \vdots \\ x_k[N-1] \\ x_k[0] \\ \vdots \\ x_k[N-1] \end{bmatrix}_{(N+P)\times 1} = \begin{bmatrix} 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}_{(N+P)\times N} \cdot x_k = \begin{bmatrix} 0_{P\times(N-P)} & I_P \\ \multicolumn{2}{c}{I_N} \end{bmatrix} \cdot x_k$$

$$y_k = A_{(N+P)\times N} \cdot x_k$$

$$t = [y_0^T \quad \cdots \quad y_{M-1}^T]^T = \begin{bmatrix} y_0 \\ \vdots \\ y_{M-1} \end{bmatrix} = \begin{bmatrix} A_{(N+P)\times N} \cdot x_0 \\ \vdots \\ A_{(N+P)\times N} \cdot x_{M-1} \end{bmatrix} = \begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ \vdots \\ x_{M-1} \end{bmatrix}$$

$$= \begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix} \cdot \begin{bmatrix} F' \cdot s_0 \\ \vdots \\ F' \cdot s_{M-1} \end{bmatrix} = \begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix}_{M\times M} \cdot \begin{bmatrix} F' & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & F' \end{bmatrix}_{M\times M} \cdot s$$

Note: the command of constructing a block diagonal matrix in Matlab 2009a (64-bit) is *blkdiag*

$$let\ U = blkdiag\{F' \quad \cdots \quad F'\} = \begin{bmatrix} F' & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & F' \end{bmatrix}_{M\times M}$$

$$let\ G = blkdiag\{A \quad \cdots \quad A\} = \begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix}_{M\times M}$$

$$M = \begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix}_{M\times M} \cdot \begin{bmatrix} F' & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & F' \end{bmatrix}_{M\times M}$$

2. Of course, your matrix multiplication description of the transmitter operation is useful for analytical purposes alone, since there is a far more efficient way to compute t from s. Describe such an efficient means of computing **t** from s by computing the inverse DFT and the addition of the cyclic prefix more efficiently.

In order to compute **t** from **s** more efficiently, we could follow these two steps:
Step 1: A N-point IFFT could be used to calculate x_k from s_k
Step 2: The addition of the cyclic prefix could be done by simply appending the last P elements of x_k to its beginning instead of multiplying by matrix G.

3. Write a Matlab program that computes t from s in the efficient manner from the previous problem.

The code is in section 3 "Putting It All Together", see "Transmitter" part.

2.1 Channel

The output of the transmitter is modulated using quadrature amplitude modulation (which you encountered in the last project) and then transmitted over a multipath wireless communications channel. We will use a simple model for the wireless communications channel as convolution of t with a causal finite impulse response filter h of length L shown in Figure 2.

1. Letting the elements of **h** be denoted by h[0], h[1],..., h[L −1], write the relation between the input to the communications channel **t** and the output from the communications channel r as a matrix multiplication. Specify the elements of the matrix.

$$r = h * t$$

$$r[n] = \sum_{k=0}^{L-1} h[k]\, t[n-k]$$

$$r[0] = h[0] \cdot t[0]$$
$$r[1] = h[1] \cdot t[0] + h[0] \cdot t[1]$$
$$r[2] = h[2] \cdot t[0] + h[1] \cdot t[1] + h[0] \cdot t[2]$$
$$\vdots$$
$$r[M \cdot (N+P) + L - 1] = h[L-1] \cdot t[M \cdot (N+P)]$$

$$r = \begin{bmatrix} h[0] & 0 & 0 & \cdots & 0 & 0 & 0 \\ h[1] & h[0] & 0 & \cdots & 0 & 0 & 0 \\ \vdots & h[1] & h[0] & \cdots & \vdots & \vdots & \vdots \\ h[L-1] & \vdots & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & h[L-1] & & \ddots & h[0] & 0 & 0 \\ \vdots & \vdots & \ddots & & h[1] & h[0] & 0 \\ 0 & 0 & \cdots & h[L-1] & \cdots & h[1] & h[0] \end{bmatrix} \cdot t$$

$$r_{(M\cdot(N+P)+L-1)\times 1} = C_{(M\cdot(N+P)+L-1)\times M\cdot(N+P)} \cdot t_{M\cdot(N+P)\times 1}$$

2. Which is a more computationally efficient manner for computing r from t in a Matlab simulation?
   (a) Using the command **convmtx** to create a convolution matrix from **h**, and then multiply t by that matrix, or
   (b) Using the command **conv** to convolve **t** with **h**?
   Why?

   conv is the more computationally efficient manner for computing r from t. This is because in convmtx, multiplication with 0 waste a lot of time.

3. Write a line of MATLAB code that simulates the operation of the channel in producing r from t.

   The code is in section 3 "Putting It All Together", see "Channel" part.

4. As we will see in the next section, it is desirable for the channel impulse response h to have a small length L, or at least have most of its energy concentrated at a few taps near zero delay. Of all channels with the same magnitude spectrum as h, which one is then the most desirable?

   The channel of linear phase is the most desirable one. Because it generates the same delay for any frequencies.

2.2 Receiver

The output of the communications channel is next passed through a receiver chain pictured in 3. The receiver first converts the signal r to M blocks of length P +N which we call v_k, $k \in \{0,..., M-1\}$, removes the cyclic prefix symbols from each block to get w_k, $k \in \{0,..., M-1\}$each of length N. Next the receiver computes the length N DFT of each block w_k to get z_k, $k \in \{0,..., M-1\}$.

1. Write the relationship between r and $z := [z_0, z_1,..., z_{M-1}]$ as a matrix multiplication. You may write the matrix involved as the multiplication of two matrices, on which describes the removal of the cyclic prefix and another which describes the operation of the blockwise DFT. Be sure to specify the elements of the matrices.

$$w_k = \begin{bmatrix} v_k[P] \\ \vdots \\ v_k[P+N-1] \end{bmatrix} = \left[\begin{matrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0_{N\times P} \end{matrix}\middle|\begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1_{N\times N} \end{matrix}\right] \cdot \begin{bmatrix} v_k[0] \\ v_k[1] \\ \vdots \\ v_k[P+N-1] \end{bmatrix}$$

$$w_k = R \cdot v_k$$

$$R = [0_{N\times P} | I_{N\times N}]_{N\times(N+P)}$$

$$z_k = DFT(w_k) = \frac{1}{\sqrt{N}} \begin{bmatrix} \omega_N^0 & \cdots & \omega_N^{0\cdot(N-1)} \\ \vdots & \ddots & \vdots \\ \omega_N^{(N-1)\cdot 0} & \cdots & \omega_N^{(N-1)\cdot(N-1)} \end{bmatrix} \cdot \begin{bmatrix} w_k[0] \\ \vdots \\ w_k[N-1] \end{bmatrix}$$

$$z_k = F_{N\times N} \cdot w_k$$

$$F_{N\times N} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \cdots & \omega^{(N-1)} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{(N-1)} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}_{N\times N}$$

$$F_{ij} = \frac{1}{\sqrt{N}} \omega_N^{(i-1)(j-1)}, i \in \{1,2,3,\dots,N\}, j \in \{1,2,3,\dots,N\}$$

$$z = \begin{bmatrix} F & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & F \end{bmatrix}_{M\times M} \cdot \begin{bmatrix} R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R \end{bmatrix}_{M\times M} \cdot v$$

2. What is a more efficient way of computing z from r than the matrix multiplication you described above? You answer should provide an alternative to matrix multiplication for both the DFT operation and the cyclic prefix removal.

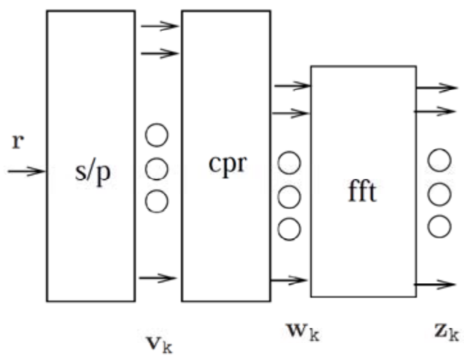


Figure 3: Part of a simple multicarrier communications receiver.

In order to compute **z** from **r** more efficiently, we could follow these two steps:
Step 1: The removal of the cyclic prefix could be done by simply deleting the first P elements of every v_k or just save the last N elements of every v_k.
Step 2: A N-point FFT could be used to calculate w_k from z_k

3. Write some Matlab code which can compute z from r in the efficient manner you just described.

The code is in section 3 "Putting It All Together", see "Receiver" part.

4. Under what condition between P and L can we write w_k as depending on only the kth block of s?

$$P \geq L - 1$$

otherwise $s_k[0]$ will include terms from the previous block, i.e. there will be aliasing between blocks.

5. Write the definition of a circulant matrix and cite your source. Suppose the conditions from the previous problem were satisfied, can you write **w** as the multiplication of $\mathbf{x_k}$ with a circulant matrix Q whose elements are only elements of h and zeros? If so, define the elements of Q. What familiar operation does multiplication with Q perform?

$$\textit{An } n \times n \textit{ matrix } C \textit{ of the form}$$

$$C = \begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix}$$

$$\textit{is called a circulant matrix.}$$

Toeplitz and Circulant Matrices: A Review, by R. M. Gray
(http://www-ee.stanford.edu/~gray/toeplitz.pdf)

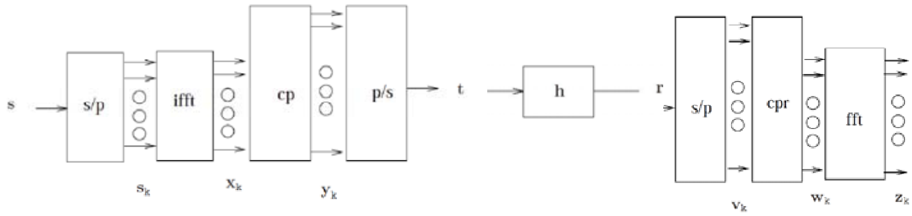
$$w_k = Q \cdot x_k$$

$$Q = \begin{bmatrix} h[0] & 0 & \cdots & 0 & h[L-1] & \cdots & h[1] \\ h[1] & h[0] & \cdots & 0 & 0 & \cdots & h[2] \\ \vdots & h[1] & h[0] & \cdots & & & \vdots \\ h[L-1] & \vdots & \vdots & \ddots & 0 & & \\ 0 & h[L-1] & & & h[0] & 0 & \\ \vdots & \vdots & \ddots & \vdots & & h[0] & 0 \\ 0 & 0 & \cdots & h[L-1] & \cdots & h[1] & h[0] \end{bmatrix}_{N \times N}$$

multiplication with Q is similar to circular convolution $x_k \quad h$.

6. Suppose again that the condition you derived in problem 4 were satisfied.

(a) Which element(s) of s_k does the ith element of z_k depend on?



for the whole signal:

$$t = \begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix}_{M\times M} \cdot \begin{bmatrix} F' & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & F' \end{bmatrix}_{M\times M} \cdot s$$

$$r = \begin{bmatrix} h[0] & 0 & 0 & \cdots & 0 & 0 & 0 \\ h[1] & h[0] & 0 & \cdots & 0 & 0 & 0 \\ \vdots & h[1] & h[0] & \cdots & \vdots & \vdots & \vdots \\ h[L-1] & \vdots & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & h[L-1] & & \ddots & h[0] & 0 & 0 \\ \vdots & \vdots & \ddots & & h[1] & h[0] & 0 \\ 0 & 0 & \cdots & h[L-1] & \cdots & h[1] & h[0] \end{bmatrix} \cdot t$$

$$z = \begin{bmatrix} F & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & F \end{bmatrix}_{M\times M} \cdot \begin{bmatrix} R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R \end{bmatrix}_{M\times M} \cdot r$$

for each block:

$$t' = A \cdot F' \cdot s_k$$
$$r' = C \cdot t'$$
$$z_k = F \cdot R \cdot r'$$

write everything together:

$$z_k = F \cdot R \cdot C \cdot A \cdot F' \cdot s_k$$
$$z_k = F \cdot Q \cdot F' \cdot s_k$$

in fact, $F \cdot Q \cdot F'$ will be a $N \times N$ diagonal matrix whose diagonal entries are the N-point DFT of $h[n]$

$$z_k = \begin{bmatrix} H[0] & 0 & \cdots & 0 \\ 0 & H[1] & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & H[N] \end{bmatrix}_{N\times N} \cdot s_k \qquad (6.a)$$

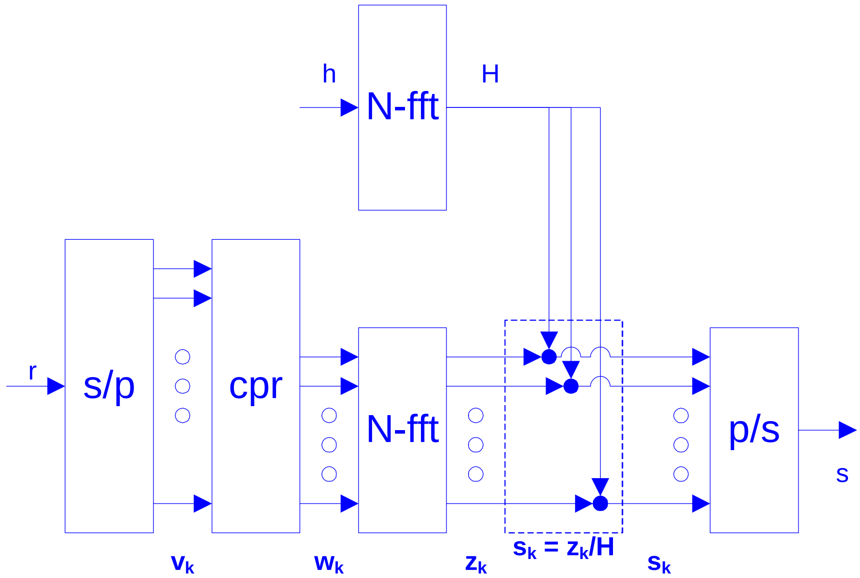
based on (6.a), obviously, the ith elements of z_k depend on the ith element of s_k.

(b) How may you reconstruct s_k from z_k if we know the channel h? We call this operation equalization.

$$s_k = \begin{bmatrix} H[0] & 0 & \cdots & 0 \\ 0 & H[1] & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & H[N] \end{bmatrix}_{N\times N}^{-1} \cdot z_k$$

$$s_k = \begin{bmatrix} H[0]^{-1} & 0 & \cdots & 0 \\ 0 & H[1]^{-1} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & H[N]^{-1} \end{bmatrix}_{N\times N} \cdot z_k$$

(c) Re-draw the receiver block diagram to include the correction for the effects of the channel that you determined in the previous problem.



(d) Implement the reconstruction of $\mathbf{s}_k$ from $\mathbf{z}_k$ using the knowledge of h with MATLAB code.

The code is in section 3 "Putting It All Together".

(e) What property of the DFT are we using in the previous problem to correct the effects of the channel h on the message signal s?

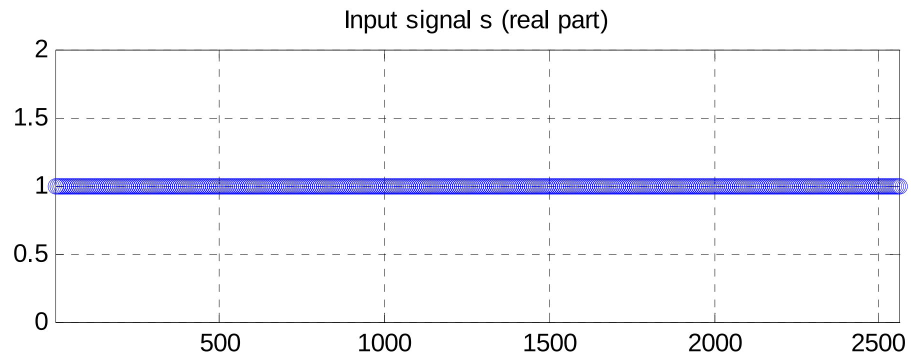
The circular convolution property.

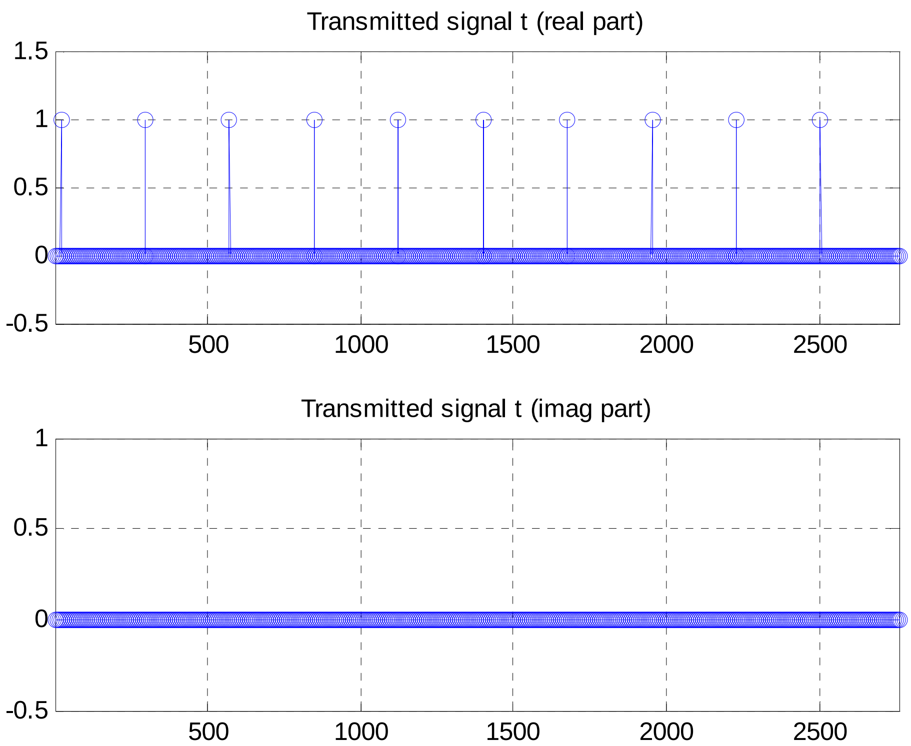
3. Putting It All Together

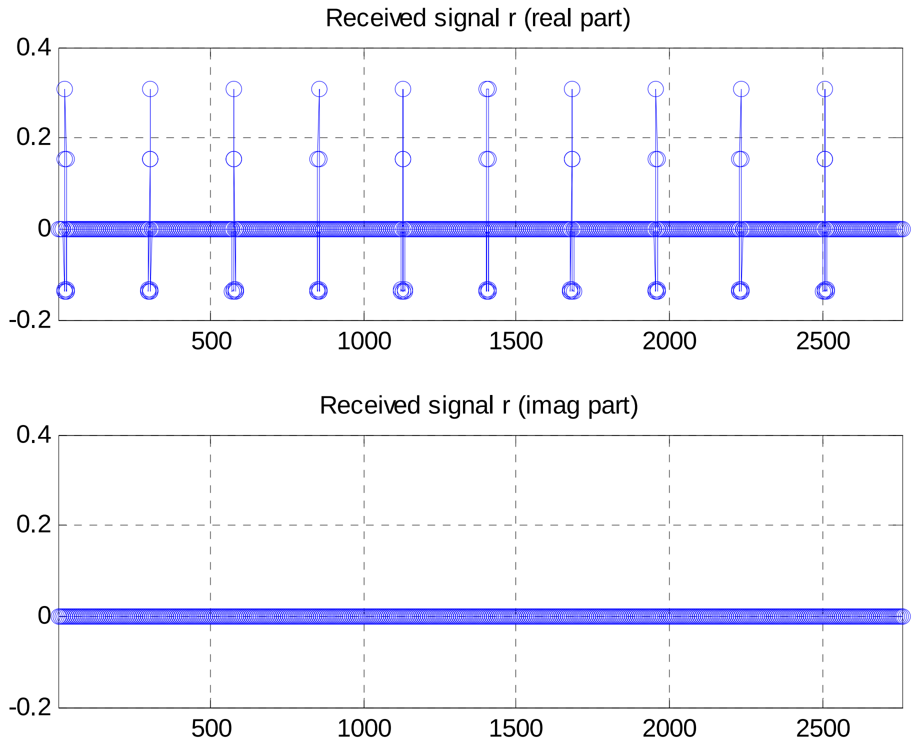
Collect all of your MATLAB code into a single file which implements a simulation of the transmitter, channel, and receiver all together. Demonstrate the proper reconstruction of the original message from the input to the transmitter at the output of the receiver by choosing s to be a vector filled with 1s and by plotting the output of your equalizer which reconstructs s from z.

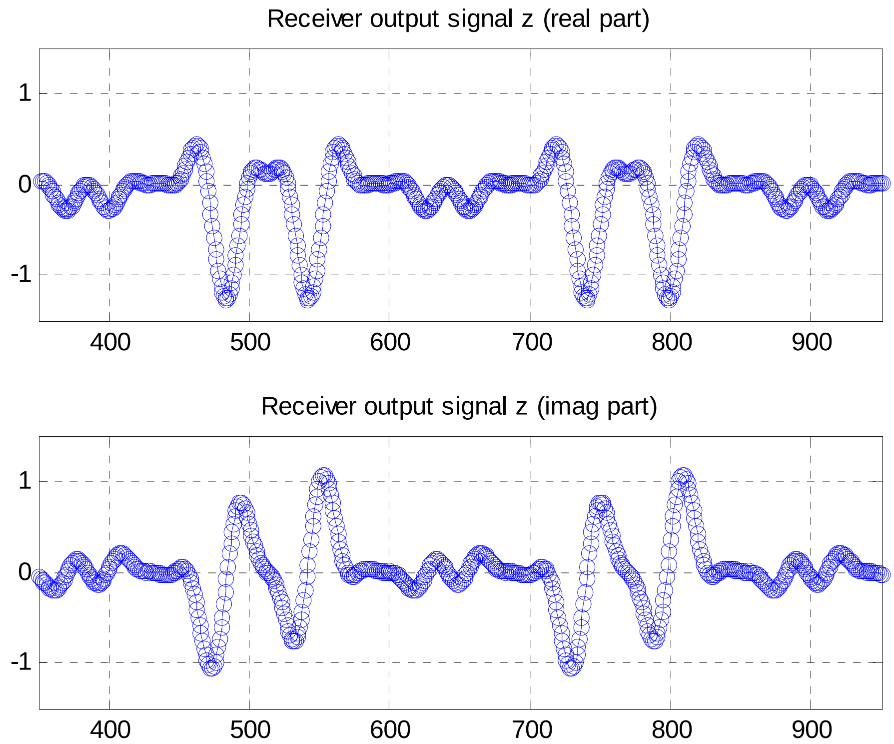
```
clear;
clc;
% ===============
% Input arguments
% ===============
M = 10;
N = 256;
P = 20;
L = 10;
% ============
% Input signal
% ============
s = ones(M*N, 1);
figure(1);
subplot(2,1,1);
plot(real(s),'o-'); grid on;
axis([1 M*N 0 2]);
title('Input signal s (real part)');
subplot(2,1,2);
plot(imag(s),'o-'); grid on;
axis([1 M*N -1 1]);
title('Input signal s (imag part)');
% ================================
% Transmitter - Serial to Parallel
% ================================
sp = zeros(N, M);
for i = 1:M
   sp(:,i) = s((i-1)*N+1:i*N);
end
% ==================
% Transmitter - IFFT
% ==================
x = zeros(N, M);
for i = 1:M
   x(:,i) = ifft(sp(:,i), N);
end
% ===========================
% Transmitter - cyclic prefix
% ===========================
y = zeros(N+P, M);
for i = 1:M
   y(:,i) = [x(N-P:N-1,i);x(:,i)];
end
% ================================
% Transmitter - Parallel to Serial
% ================================
t = zeros(M*(N+P), 1);
for i = 1:M
   t((i-1)*(N+P)+1:i*(N+P)) = y(:,i);
end
figure(2);
subplot(2,1,1);
plot(real(t), 'o-'); grid on;
axis([1 M*(N+P) -0.5 1.5]);
title('Transmitted signal t (real part)');
subplot(2,1,2);
plot(imag(t), 'o-'); grid on;
axis([1 M*(N+P) -0.5 1]);
title('Transmitted signal t (imag part)');
% =======
% Channel
% =======
```

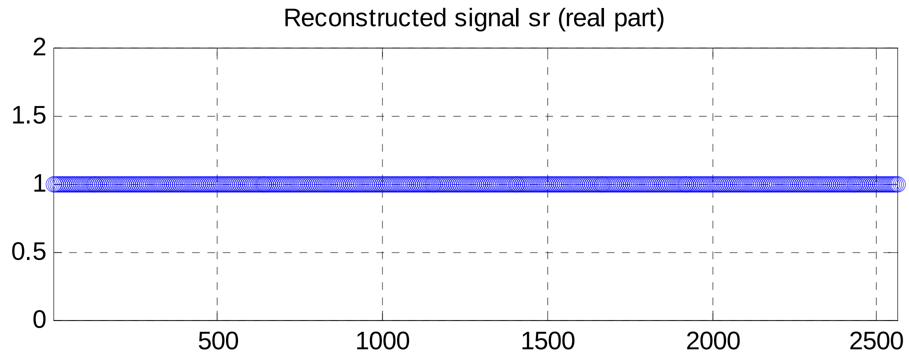
```
h = firpm( L-1, [0 1/16 1/8 3/8 7/16 1], [0,0,1,1,0,0] );
h = h';
r = conv(t, h);
r = r(1:M*(N+P));
figure(3);
subplot(2,1,1);
plot(real(r), 'o-'); grid on;
axis([1 M*(N+P) -0.2 0.4]);
title('Received signal r (real part)');
subplot(2,1,2);
plot(imag(r), 'o-'); grid on;
axis([1 M*(N+P) -0.2 0.4]);
title('Received signal r (imag part)');
% ============================
% Receiver - Serial to Parallel
% ============================
v = zeros(N+P, M);
for i = 1:M
   v(:,i) = r((i-1)*(N+P)+1:i*(N+P));
end
% ===============================
% Receiver - cyclic prefix removal
% ===============================
w = zeros(N, M);
for i = 1:M
   w(:,i) = v(P+1:N+P,i);
end
% ==============
% Receiver - FFT
% ==============
z = zeros(N, M);
for i = 1:M
   z(:, i) = fft(w(:,M), N);
end
% ============================
% Receiver - Parallel to Serial
% ============================
zs = zeros(N*M,1);
for i = 1:M
   zs((i-1)*N+1:i*N) = z(:,i);
end
figure(4);
subplot(2,1,1);
plot(real(zs), 'o-'); grid on;
axis([350 950 -1.5 1.5]);
title('Receiver output signal z (real part)');
subplot(2,1,2);
plot(imag(zs), 'o-'); grid on;
axis([350 950 -1.5 1.5]);
title('Receiver output signal z (imag part)');
% =========
% Equalizer
% =========
H = fft(h, N);
sr = zeros(M*N,1);
for i = 1:M
   sr((i-1)*N+1:i*N) = z(:,i)./H;
end
figure(5);
subplot(2,1,1);
plot(real(sr), 'o-'); grid on;
axis([1 M*N 0 2]);
title('Reconstructed signal sr (real part)');
subplot(2,1,2);
plot(imag(sr), 'o-'); grid on;
axis([1 M*N -1 1]);
title('Reconstructed signal sr (imag part)');
```

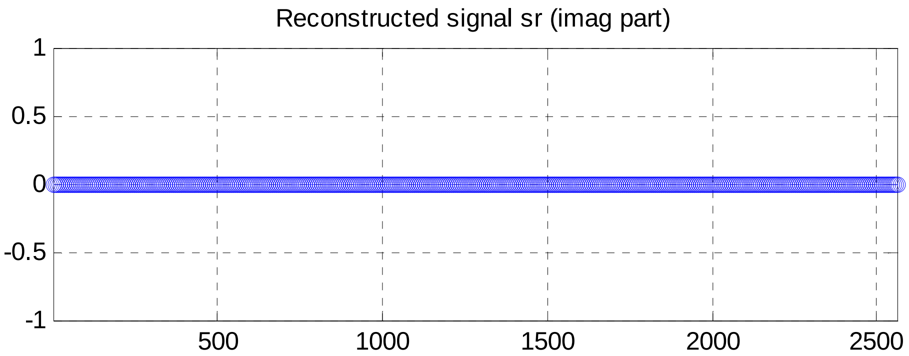

Input signal s (real part)
2
1.5
1
0.5
0
500
1000
1500
2000
2500
Input signal s (imag part)
1
0.5
0
-0.5
-1
500
1000
1500
2000
2500


Transmitted signal t (real part)
1.5
1
0.5
0
-0.5
500
1000
1500
2000
2500
Transmitted signal t (imag part)
1
0.5
0
-0.5
500
1000
1500
2000
2500


Received signal r (real part)
0.4
0.2
0
-0.2
500
1000
1500
2000
2500
Received signal r (imag part)
0.4
0.2
0
-0.2
500
1000
1500
2000
2500


Receiver output signal z (real part)
1
0
-1
400
500
600
700
800
900
Receiver output signal z (imag part)
1
0
-1
400
500
600
700
800
900


Reconstructed signal sr (real part)
2
1.5
1
0.5
0
500
1000
1500
2000
2500


Reconstructed signal sr (imag part)
1
0.5
0
-0.5
-1
500
1000
1500
2000
2500