B1 Engineering Computation Project
Spectral Shaping for Communication Systems

Introduction

With the increasing popularity of ultra-wideband communication, the interference between UWB and other narrowband transmissions has to be avoided. This report demonstrates the procedure of modifying a randomly generated signal sequence to achieve appropriate interference avoidance.

1.Generating Random Signals

The simpler code works instead of a "for" loop by generating a matrix of length N and get element values of matrix b in one command. The const_sum manipulation on a matrix is effective to the manipulation on each element of the matrix.

To make both QPSK and 16-QAM as the const str function input option, use 'if' or 'switch'.

```
if ~strcmpi(const_str,'qpsk') && ~strcmpi(const_str,'16-qam')
    error('Only QPSK or 16-QAM is supported.');
end

if strcmpi(const_str,'qpsk')==1
% QPSK constellation
const_sym = [1+1i 1-1i -1+1i -1-1i];

else
%16-QAM constellation
const_sym = [1+1i 1-1i 1+3i 1-3i -1+1i -1-1i -1+3i -1-3i 3+1i 3-1i 3+3i 3-3
```

Fig 1. The 'if' case

```
switch const_str
    case 'qpsk'
        const_sym = [1+1i 1-1i -1+1i -1-1i];
        % QPSK constellation
    case '16-qam'
        %16-QAM constellation
        const_sym = [1+1i 1-1i 1+3i 1-3i -1+1i -1-1i -1+3i -1-3i 3+1i 3-1i
    otherwise
        error('Only QPSK or 16-QAM is supported.');
end
```

Fig 2. The 'switch' case

```
% Initialise vector for storing symbols
d = zeros(N,1);


% Randomly assign constellation symbols to vector
for i = 1:N
    % 1st argument of randi: maximum integer in range
    % 2nd argument: 1 specifies a scalar output
    sym_index = randi(M,1);
    d(i) = const_sym(sym_index);
end
```

Fig 3. Method1 for assigning signal vector 'd'

```
M = length(const_sym);  % Get the number of constellation points
% Randomly assign constellation symbols to vector
sym_indices = randi(M,[1 N]);  % Note the additional argument
d = const_sym(sym_indices);    % Directly construct the N-tuple
d = d.';

end
```

Fig 4. Method2 for assigning signal vector 'd'

Here 'd' is not initialized in Fig 4 because it's assigned to a [1 N] matrix at one time.

2. Plotting Generated Signals

(With vertical axis as imaginary axis and horizontal axis as real axis in the following graphs)

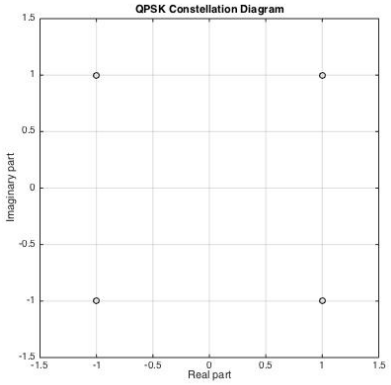


Fig 5. QPSK constellation

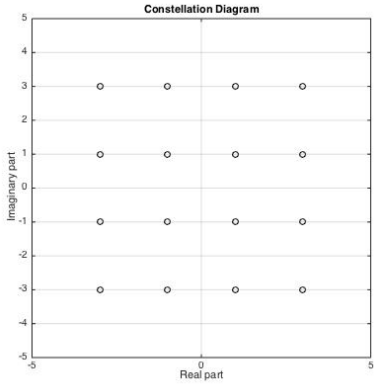


Fig 6. 16-QAM constellation

3. Spectral Analysis

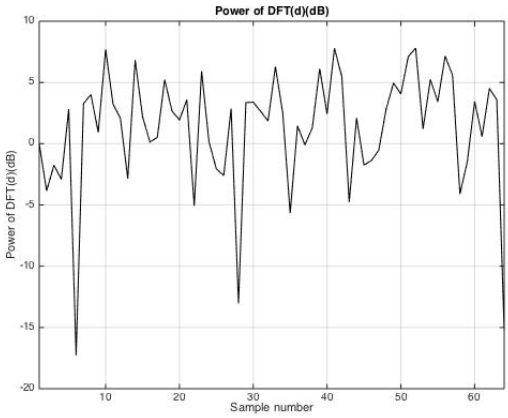


Fig 7. Discrete fourier transform of generated QPSK signal in dB with N = 64

4. Construct Transfer Function (Window)

Objective: To create a deep notch in the data signal

The Naive Approach: Simply nulling the required frequency (here is the indices) components of DFT of d and transfer it back to time domain.

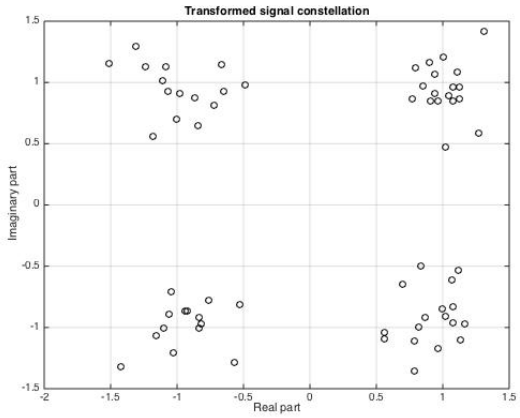


Fig 8. Constellation diagram of transformed QPSK signal with N = 64 and indices [9:12]

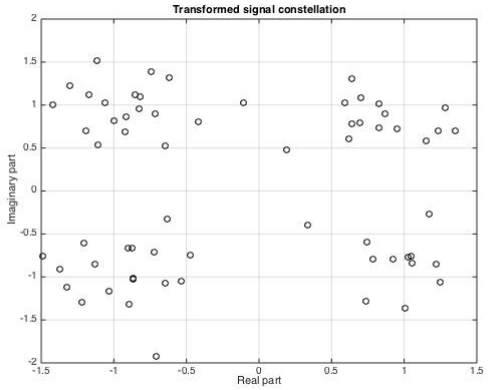
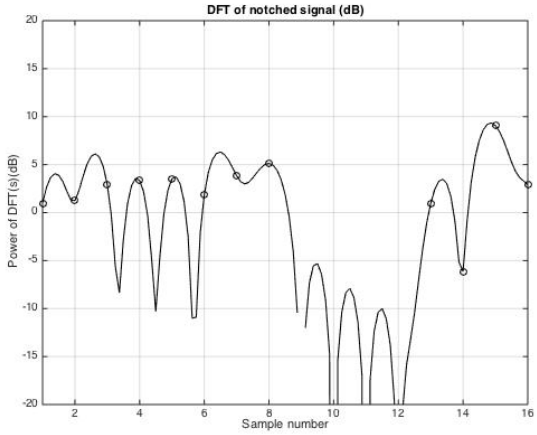


Fig 9. Constellation diagram of transformed QPSK signal with N = 64 and indices [9:18]

Simply nulling causes both scaling and phase-shift on the original signal. The constellation graph with indices [9:18] if more spread, indicating that with simply nulling, the larger the notch region is relative to the original signal, the more the signal is damaged. Simple notch cause the data in the notched interval lost forever.

5. Optimization

Upsampling (interpolating points between original notched time-domain response)



The usage of upsampling approximates the sequence at a higher sampling frequency than the actual frequency, which gives a better approximation of the continuous signal.

Fig 10. Upsampling of DFT of notched signal (dB)

a. The Eigen-Window Method

Window coefficients are set to real numbers, causing the phase transfer function to be unity. The phase information of the signal is remained intact. Thus, we are expecting a constellation graph with un-altered phase and a changed magnitude.

According to the definition of DFT operator matrix, W and WI(operator of the notched region) can be constructed. Then get the smallest eigenvalue as well as its corresponding eigenvector, which is the window required.

```
% Construct matrices WI, D, etc
for a=1:rho*N
    for b=1:N
        W(a,b) = 1/sqrt(N)*exp((-1i)*2*pi*a*b/(rho*N));
    end
end
WI = W(Q,:);
D = diag(d);
```

Fig 11. DFT operator construction

```
% Get window vector and normalise to meet constraints
A = inv(D'*D)* real(D'* WI'* WI * D);
[V,~] = eigs(A,N);
x = V(:,N);
r = norm(D*x)/N^(0.5);
x = x/r;
```

Fig 12. Eigen-window construction

Results:

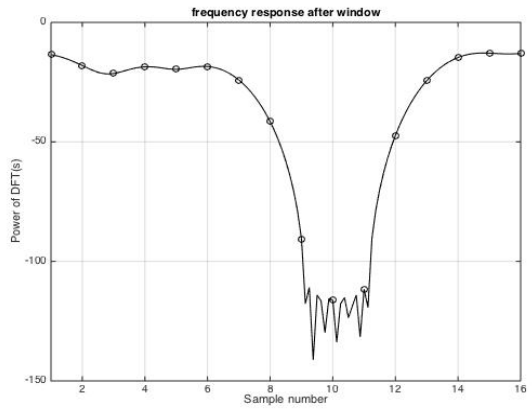


Fig 13. DFT (dB) of QPSK signal after transformed by eigen-window function With N=16. rho=8. indices=[9:11]

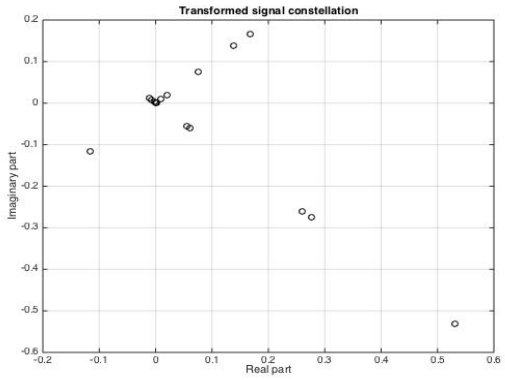


Fig 14. Constellation graph of QPSK signal after transformed by eigen-window function With N=16, rho=8, indices=[9:11]

The transformed signal forms a cross in constellation graph. The constellation points are lined up into two lines perpendicular to each other forming 45-degree angles with the axes, meaning that the only effect window has on the original signal is scaling.

Although it gives a good deep notch with eigen-window method, it's not suitable here because the elements of the window vector can be either positive or negative numbers, resulting in sign ambiguity, which has to be further decoded with knowledge of the actual window in order to get the initial signal.

b. The Barrier Method

Reformulation of the constraints can solve the sign-ambiguity problem. Also, a simple non-negative scaling on all data points will give a spare space in the center of the constellation graph so that the receiver can distinguish between constellation points without knowledge of the window.

The main disadvantage of this formulation is that this is only useful for constant-modulus signals because elements of window can be any number bigger than delta, arising a problem of magnitude ambiguity if there are signals in the same phase but different in modulus.

```
% Get window vector
p = N + 1;

x = 0.9 * ones([N,1]);
l = 0;
while p/t >= epsilon0
    l = l + 1;
    [xopt] = newton(x,epsiloni,D,WI,N,t);
    t = mu * t;
    x = xopt;  % start from new x to search the next x with new t
end
```

Fig 15. Computing optimal window x with increasing value of t until the approximation is reasonable

```
% newton's method to find optimal x with specific t
while omegasq/2 >= epsiloni
    [y1,y2] = dfx( x,D,WI,N,t,delta );
    deltx = -inv(y2)*y1;
    omegasq = -y1.' * deltx;

    %line search to determine step size (secure convergence)
    [y_new] = fx(x+theta*deltx,D,WI,N,t,delta);
    [y_old] = fx(x,D,WI,N,t,delta);
    while y_new > y_old + alpha*theta*y1.'*deltx
        theta = beta*theta;
        [y_new] = fx(x+theta*deltx,D,WI,N,t,delta);
        [y_old] = fx(x,D,WI,N,t,delta);
    end

    %compute new x
    x = x + theta * deltx;
end
```

Fig 16. Newton's method with backtracking line search to find optimal window x with current t

```
function [y] = fx(xi,D,WI,N,t,delta)

e = eye(N);
%cost function value
if N - xi.'*xi < 0
    y = Inf;
else
    y = xi.'*D'*WI'*WI*D*xi - 1/t * log(N - xi.'*xi);
    for i=1:N
        if e(:,i).'*xi - delta < 0
           y = Inf;
           break
        else
            y = y - 1/t * log(e(:,i).'*xi - delta);
        end
    end
end
end
```

Fig 17. Computation of the approximation of cost function with current t

Note:
When computing the approximate cost function (Fig 17), the log function may give complex numbers if the number is negative, causing problem when we compare cost function values in line search. This can be avoided by directly setting the function value to infinity if input value for log is negative.

Results:
Use the fast convergence initial conditions for newton's method [1] , the following graphs are generated:

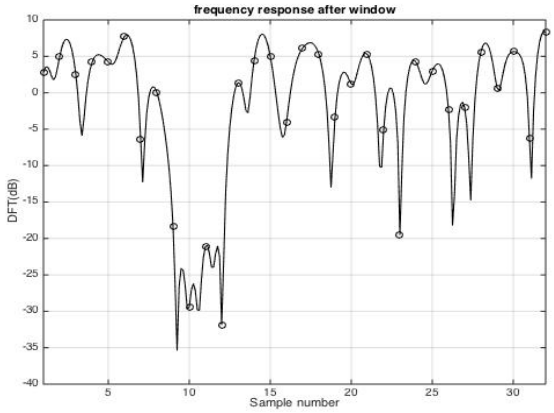


Fig 18. DFT(dB) of QPSK signal after transformed by a window function optimized by barrier method
N=32, rho=8, indices=[9:15], mu=1.5,
epsilon0=0.01,epsilon=0.5,
initial t=1

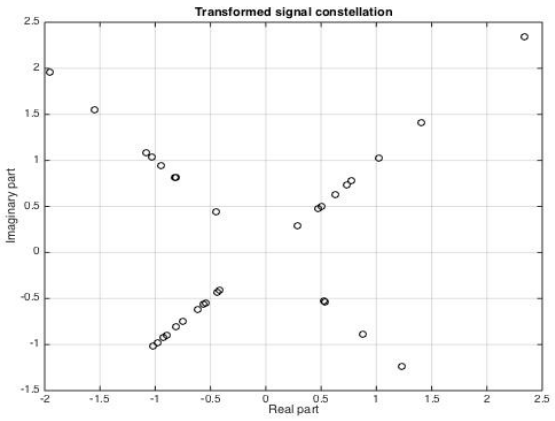


Fig 19. Constellation graph of QPSK signal after transformed by a window function optimized by barrier method
N=32, rho=8, indices=[9:15], mu=1.5, epsilon0=0.01,epsilon=0.5, initial t=1

Comments:
Barrier method gives a sufficient deep notch in the required interval and also a clear spare space in the center of constellation diagram for receiver to easily read the original signal.

6. Run-time Analysis

To analyze the run time and number of iterations of Barrier method with different parameters (epsilon0, epsiloni, initial t, and mu), a separate script is written to plot the above relations. Each graph is plotted after the generation of a new set of signal data, but signal data remains unchanged during the increasing parameter process.

Results:
Cputime (left axis)/number of iterations of t (right axis) vs. parameters

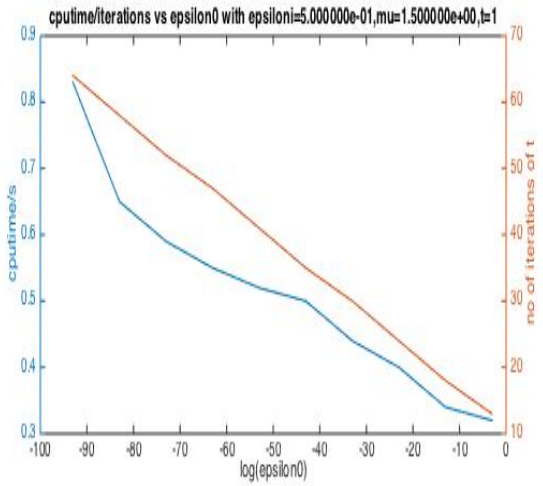


Fig 20. cputime/iterations vs. epsilon0

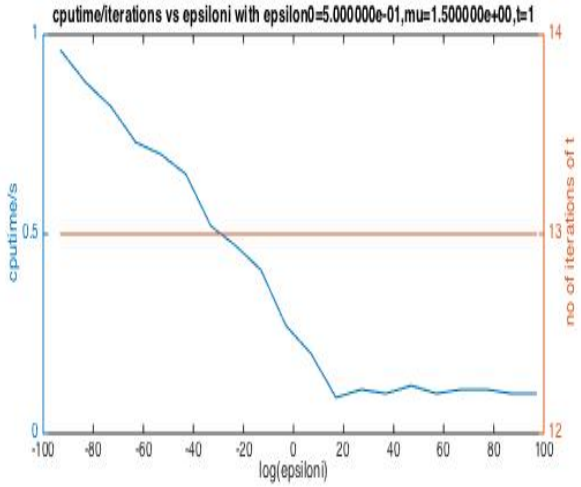


Fig 21. cputime/iterations vs. epsiloni

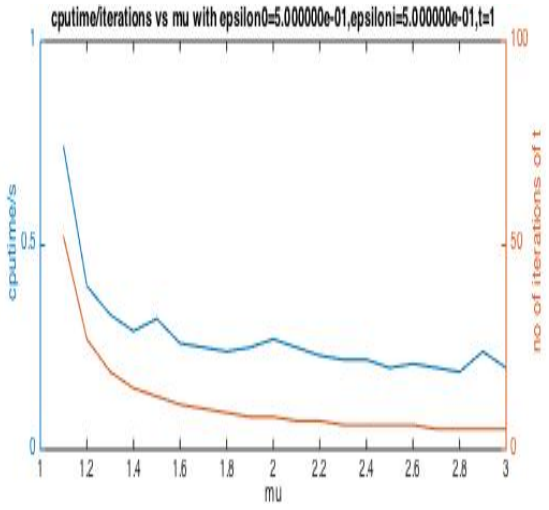


Fig 22. cputime/iterations vs. mu

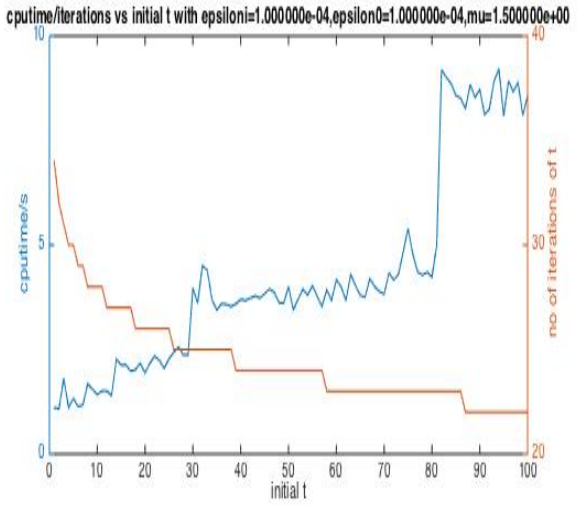


Fig 23. cputime/iterations vs. initial t

Comments:
As expected, the number of iterations of t would decrease with increase of epsilon0 or t or mu since computation continues while p/t < epsilon0 and new t = mu*t. It would not be affected by epsilon, which is used to define the accuracy of the approximation of optimal window x.

The increase of both epsilon0 and epsiloni cause a decrease in run-time because increase in tolerance means the decrease in accuracy therefore fewer steps is needed. The flat region in epsiloni graph indicates that the tolerance is too large to have any impact on the accuracy. It always meets the condition so it never gets into the accuracy measurement loop.

Cputime interestingly increase in a step shape with increasing initial t. This means that although an increase in t always tends to decrease the iteration number but for each iteration it spends even more time to compute an optimal window. In fact, larger t gives a slower change in gradient and hessian thus gives a slower approach to omega^2 < epsiloni.

7. 16-QAM Responses and Multiple Notches

Improvements can be made with Barrier method by simply reformulate the problem. To avoid modulus ambiguity problem raised with 16-QAM signals, restrict the window elements within certain region so that the region displaying small modulus signals does not overlap with that displaying large modulus. To evaluate the relationship between maximum and minimum value of x, set $\delta \leq x \leq \gamma$. In the 16-QAM case only the 2 types of signals on the diagonal of constellation diagram need to be separated, one with modulus $\sqrt{2}$, which would appear in the region of $\delta\sqrt{2} \leq r \leq \gamma\sqrt{2}$, another group with modulus $3\sqrt{2}$, which appear in $3\delta\sqrt{2} \leq r \leq 3\gamma\sqrt{2}$, with r denotes the radius on constellation graph. Therefore, $3\delta\sqrt{2} > \gamma\sqrt{2}$, giving $3\delta > \gamma$.

To achieve multiple notches, rethink the problem as minimizing the sum of the frequency responses of all notches. Write each notch region as an additional term similar to the previous single-notch problem. Here, take double notches as example.

Problem reformulation:

$$\begin{aligned} \text{Minimize} \quad & ||\boldsymbol{WI_1Dx}||_2^2 + ||\boldsymbol{WI_2Dx}||_2^2 \\ \text{Subject to} \quad & ||\boldsymbol{x}||_2^2 \leq \boldsymbol{N} \\ & \boldsymbol{\delta \leq x_i \leq \gamma} \end{aligned}$$

$\boldsymbol{WI_1}$ and $\boldsymbol{WI_2}$ are DFT operators corresponding to two notch regions. Change the corresponding cost function values and its derivatives. Choose δ and γ so that all the constraints are meet and they are as far as possible to each other allow deep notches, which in this case, theoretically, is to choose δ close to 1 (smaller than 1) and take γ slightly smaller than 3* δ. However, since the norm of window is also limited by N, in order to give x enough space to stretch, there may be a trade off between these two constraints, drawing back the value of δ to small.

Here are some examples:
Code:

```
% Get parameters
N = length(d);
i1 = min(indices1);
i2 = max(indices1);
i3 = min(indices2);
i4 = max(indices2);
% Find upsampled indices
Q1=(i1-1)*rho+1:(i2-1)*rho+1;
Q2=(i3-1)*rho+1:(i4-1)*rho+1;
% Construct matrices WI, D, etc
for a=1:rho*N
    for b=1:N
        W(a,b) = 1/sqrt(N)*exp((-1i)*2*pi*a*b/(rho*N));
    end
end

WI1 = W(Q1,:);
WI2 = W(Q2,:);
D = diag(d);
```

Fig 24. Get WI_1 and WI_2

```
function [y] = fx_16qam(xi,D,WI1,WI2,N,t,delta,gama)

e = eye(N);
%cost function value

if N - xi.'*xi < 0
    y = Inf;
else
    y = xi.'*D'*WI1'*WI1*D*xi - 1/t * log(N - xi.'*xi) + xi.'*D'*WI2'*WI2*D*xi;
    for i=1:N
        if e(:,i).'*xi - delta < 0 || gama - e(:,i).'* xi < 0
            y = Inf;
            break
        else
            y = y - 1/t * log(e(:,i).'*xi - delta) - 1/t * log(gama - e(:,i).'*xi);
        end
    end
end
end
```

Fig 25. Construct cost function

```
function [y1,y2] = dfx_16qam( x,D,WI1,WI2,N,t,delta,gama)
%y1 : first derivative
%y2 : second derivative (hessian)

% compute derivatives

e = eye(N);
y1 = 2 * real(D' * WI1' * WI1 * D) * x + (1/t) * (2/(N-x.'*x)) * x + 2 * real(D' * WI2' * WI2 * D) * x ;

y2 = 2 * real(D' * WI1' * WI1 * D) + 2/t *  (2*x*x.' + (N - x.'*x)*e)/(N - x.'*x)^2 + 2 * real(D' * WI2' * WI2

sum_y1 = zeros(N,1);
sum_y2 = 0;
for i = 1:N
    sum_y1 = sum_y1 + (1/(e(:,i).'*x-delta))*e(:,i) + (1/(e(:,i).'*x-gama))*e(:,i);
    sum_y2 = sum_y2 + e(:,i)* e(:,i).'/(e(:,i).'*x-delta)^2 + e(:,i)* e(:,i).'/(e(:,i).'*x-gama)^2;
end
y1 = y1 - 1/t * sum_y1;
y2 = y2 + 1/t * sum_y2;
```

Fig 26. Construct gradient and hessian

Results:

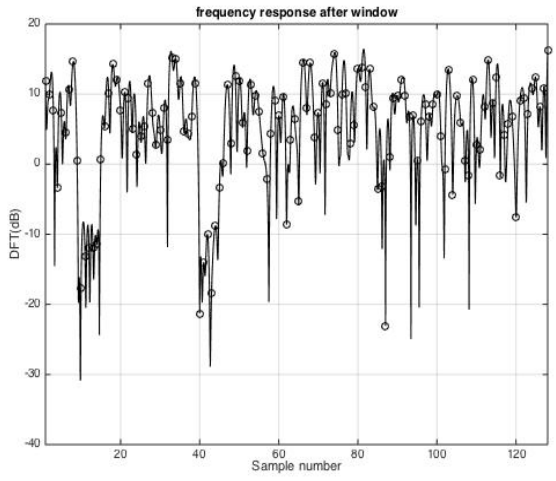


Fig 27. Frequency response with N=128

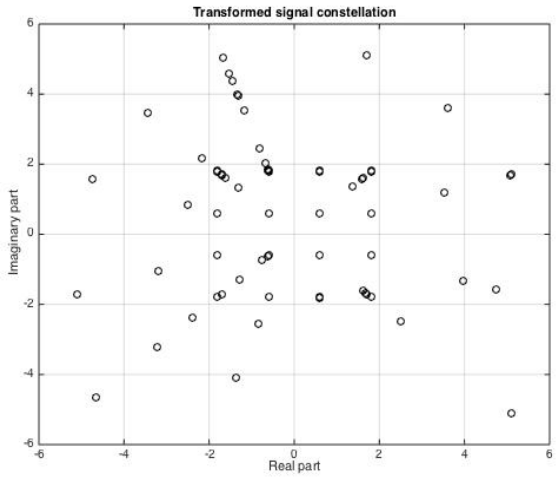


Fig 28. Constellation graph with N=128

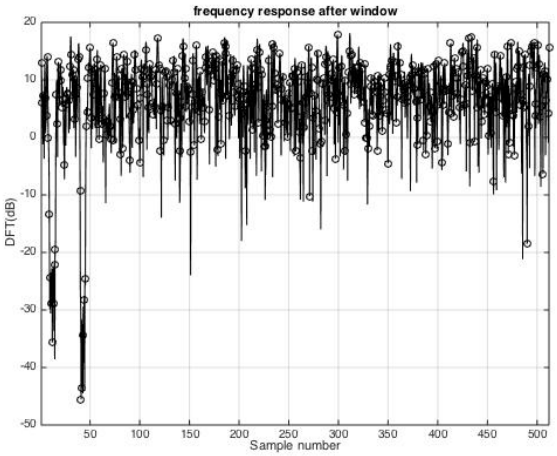


Fig 29. Frequency response with N=512

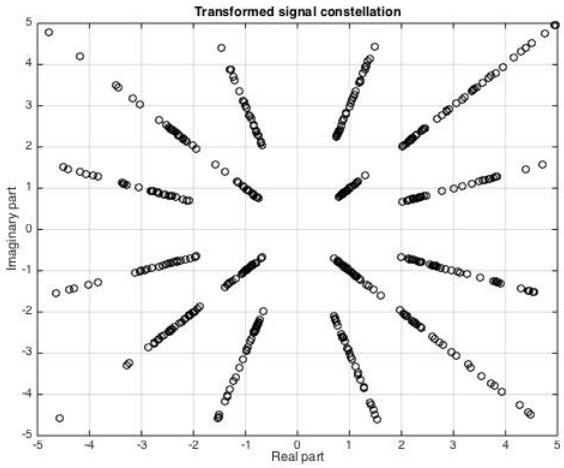


Fig 30. Constellation graph with N=512

Various values of δ and γ are tried with γ ranges from 0.5 to 2.3 and δ from 0.2 to 0.8. Most of the pairs give good notch with N=512, sometimes with N=256, rarely with N=128. It's more likely to give good results with δ in the range of 0.35 to 0.6.

We can see the discontinuity formed on the diagonals (Fig 28), indicating the success of separating signals by there modulus. The receiver can therefore easily distinguish between signals with different modulus by simply detecting its transformed modulus.

Conclusion

In this project, two types of random signals are firstly generated, with QPSK of constant modulus and 16-QAM of non-constant modulus. Then, ways of analyzing and plotting the signals in both time domain and frequency domain are explored. The problem is approached initially with simple nulling, which causes large distortion of signal phase and amplitude. Large amount of information is lost during the process. By applying a proper window vector, adding constraints requiring its elements to be real or positive, information can be remained intact. Both Eigen-window method (using Lagrange multiplier) and Barrier method (using indicator function and its approximation) can transfer constrained optimization problems into unconstrained problems. Eigen-window method successfully preserved signal's phase property, however, gives an ambiguity in sign. Barrier method gives both a deep notch in the desired interval and a good preservation of signal phase and magnitude information. It is shown that in most cases, the computation of barrier method can be done in 2 seconds for 64 QPSK signals. Other improvements can be made on Barrier method to achieve non-constant modulus signal capability and multiple notches by reformulating the optimization objective.

Reference

[1] J.P.Coon, "Narrowband interference avoidance for ultra-wideband single-carrier block transmissions with frequency-domain equalization", IEEE wireless communications, 2008