Delft University of Technology

Array Processing 1

Authors:
Maxmillan Ries (5504066) Chaufang Lin (5466091)
August 23, 2022

1 Signal Model

1.1 Task 1 - Generate

The first task of this report was generating the signal, received signal and array response matrix. Below I break down what was provided, relative to the component which needed it:

- **S**: For S, the formula of $s_{i,k} = exp(j2\pi f_i k)$ was provided, where f_i and k were provided. f_i stands for the array of frequencies for each source, and k represents the nth out of N samples.
- **A**: For A, several more components were required. Firstly, because the antennas form a uniform linear array, δ was provided. Secondly, the angle of arrival was required, provided through the array θ.
- **X**: To create X, A & S were used with only the SNR being additionally required for the awgn.

Below is a pseudocode of my work:

Algorithm 1 Generate X, A, S

procedure GENDATA($M, N, \delta, \theta, f, SNR$) ▷ Inputs provided
    Retrieve number of sources n
    S = [n × 1]
    A = [M × n]
    X = [M × N]

    for `s in S` **do** ▷ Create each source for k = 1
        `s` = $exp(j2\pi f_i)$
    end for

    for `(antenna, source) in indices(A)` **do** ▷ Array Response is the same for all samples
        A(antenna, source) = exp(j2π antenna δ sin(θ_{source}))
    end for

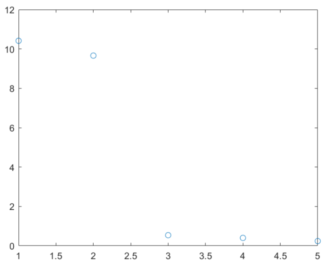
    for `k = 1:N` **do** ▷ For each sample, use $S^k = (exp(j2\pi f_i)^k) = exp(j2\pi f_i k)$
        S' = S^k
        X(:, k) = awgn((A * S'), SNR)
    end for

    return X, A, S
end procedure

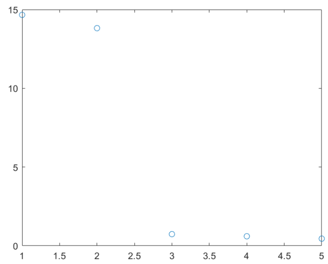
For generating the data, it would normally also be possible to create a larger **S** matrix with all the samples filled in, but the end results is the same since $(a^b)^c = a^{b*c}$.

1.2 Task 2 - Singular Values

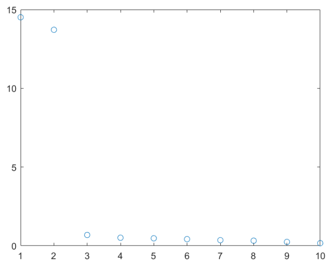
In order to discuss the changes in the singular values, I have created the following plots.



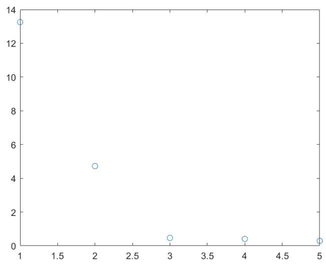
(a) Normal



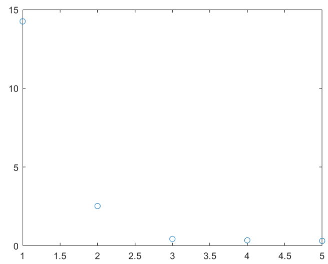
(b) Doubled Sample Count



(c) Doubled Antenna Count



(d) Close Antennas



(e) Close Frequency

Figure 1: Image showing the effect of different changes on the singular values.

What happens to the singular values if the number of samples doubles?
The ratio between the singular values representing the 2 signals and those representing the noise will increase.

What happens to the singular values if the number of antennas doubles?
The ratio between the singular values representing the 2 signals and those representing the noise will increase.

What happens to the singular values if the angles between the sources becomes small?
The singular value decomposition shows that the second source becomes harder to differentiate from noise. If the angles are the same then it appears that there is only one signal, as such, as the angle difference decreases, only one signal singular value seems to be present.

What happens to the singular values if the frequency difference comes small?
The singular value decomposition shows that the second source becomes harder to differentiate from noise. If the frequencies are the same then it appears that there is only one signal, as such, as the angle difference decreases, only one signal singular value seems to be present.

2 Estimation of Direction

Algorithm 2 ESPRIT for direction

```
procedure ESPRIT(X, d)                          ▷ Inputs provided
    X_top = Top M-1 rows of X
    X_bottom = Bottom M-1 rows of X
    Z = [X_top; X_bottom]                       ▷ Vertically concatenated
    [U,S,V] = svd(Z);                           ▷ Economic SVD used to trim out 0's
    Take d columns of U                         ▷ d = number of signals
    Ux = Top half of U
    Uy = Bottom half of U
    Values = eig(pinv(Ux) · Uy)
    for e in Values do
        θ_i = 180π · asin(angle(e)/π)
    end for
    return θ
end procedure
```

The assignment specifically asks to check that the algorithm works, and it does. For a later Task, I show the effect of noise on the ESPRIT direction estimation.

3 Estimation of Frequency

Algorithm 3 ESPRIT for frequency

procedure ESPRITFREQ(X, d) ▷ Inputs provided
    X_t = First row of X
    N = Number of samples
    m = N/2 ▷ Can be arbitrarily chosen

    Z = [m × N-m]

    for `j = 1:N-m` **do**
        $Z(:, j) = X_t(1 + (j - 1) : m + (j - 1))$
    end for

    [U,S,V] = svd(Z); ▷ Economic SVD used to trim out 0's

    Take d columns of U ▷ d = number of signals

    Ux = Top $m - 1$ of U
    Ux = Bottom $m - 1$ of U

    Values = eig(pinv(Ux) · Uy)

    for `e in Values` **do**
        f_i = angle(e) / 2π
    end for

    return f
end procedure

The assignment specifically asks to check that the algorithm works, and it does. For a later Task, I show the effect of noise on the ESPRIT frequency estimation.

4 Joint Estimation of Direction and Frequency

I will preface this section with a confession that I could not understand enough of the lecture by A.J van der Veen to complete the assignment. I did however find the master thesis of Joost Geelhoed, which helped me immensely understand how to work this out. Below is the pseudocode:

Algorithm 4 Joint Diagonalization for direction and frequency

procedure JOINT(X, d, m) ▷ Inputs provided
  Z = [m · M × N-m] ▷ M = number of antennas
  Z = $\begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{N-m} \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{N-m+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{m-1} & \mathbf{x}_m & \cdots & \mathbf{x}_{N-1} \end{bmatrix}$ ▷ **x** signifies the vector with all antennas

  [U,S,V] = svd(Z); ▷ Economic SVD used to trim out 0's

  Trim U columns based of input d

  $\delta_{\theta,top} = [I_M \ 0_1]$ ▷ Direction Estimation Component
  $\delta_{\theta,bottom} = [0_1 \ I_M]$

  $\delta_{\theta,top,m} = I_m \bigotimes \delta_{\theta,top}$
  $\delta_{\theta,bottom,m} = I_m \bigotimes \delta_{\theta,bottom}$

  Ux = $\delta_{\theta,top,m} \cdot U$
  Uy = $\delta_{\theta,bottom,m} \cdot U$

  $pUxUy_\theta$ = pinv(Ux) · Uy

  $\delta_{f,top,M} = [I_m \ 0_1] \bigotimes I_M$ ▷ Frequency Estimation Component
  $\delta_{f,bottom,M} = [0_1 \ I_m] \bigotimes I_M$

  Ux = $\delta_{f,top,M} \cdot U$
  Uy = $\delta_{f,bottom,M} \cdot U$

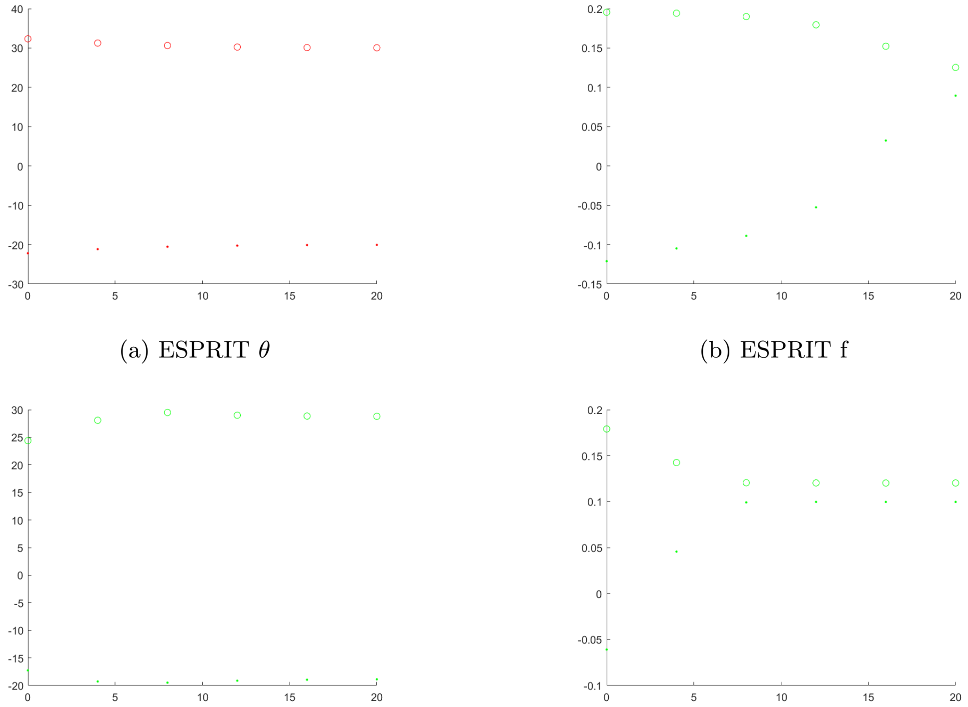
  $pUxUy_f$ = pinv(Ux) · Uy

  $[\theta,\ f]$ = JointSolver$(pUxUy_\theta,\ pUxUy_f)$ ▷ Provided by Didem

  return θ, f
end procedure

5 Comparison

5.1 Estimation Performance



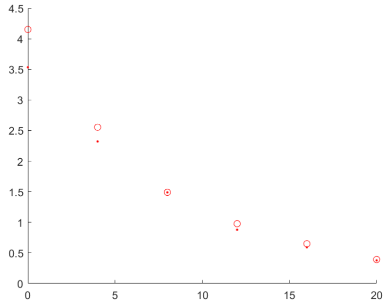
(a) ESPRIT θ

(b) ESPRIT f

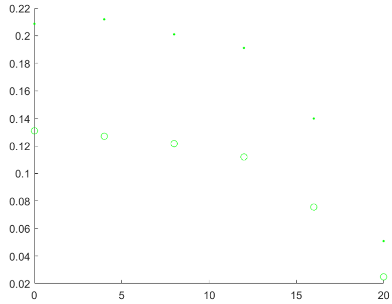
(c) JOINT θ

(d) JOINT f

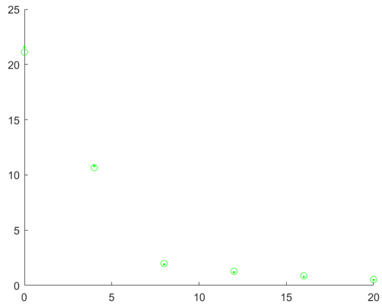
Figure 2: Plots of Means for both direction and frequency estimation



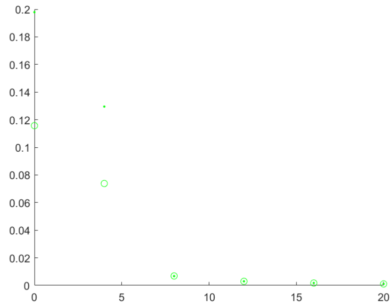
(a) ESPRIT θ



(b) ESPRIT f



(c) JOINT θ



(d) JOINT f

Figure 3: Plots of SD for both direction and frequency estimation

Generally observing the graphs, as one would expect, as the Signal to Noise Ration (SNR) increases, it becomes easier to estimate the angle and frequency using both ESPRIT and Joint Diagonalization.

5.2 Beamforming

Algorithm 5 Zero Forcing for direction

procedure ZERO_FORCING_$\theta(X, \delta, \theta)$ ▷ Inputs provided
    A = [M $\times$ n] ▷ n is the number of sources
    for `(antenna, source) in indices(A)` **do** ▷ Array Response is the same for all samples
        A(antenna, source) = exp(j2π antenna δ sin(θ_{source}))
    end for

    $w^H = (A^H \cdot A)^{-1} \cdot A^H$

    return $S_\theta = w^H \cdot X$
end procedure

Algorithm 6 Zero Forcing for frequency

procedure ZERO_FORCING_$\theta(X, f)$ ▷ Inputs provided

  S $= [n \times N]$

  for `(source, sample) in indices(S)` **do** ▷ Array Response is the same for all samples

    S(source, sample) $= \exp(j2\pi\ f\ sample)$

  end for

  $A_f = (X \cdot S^H) \cdot (S * S^H)^{-1}$

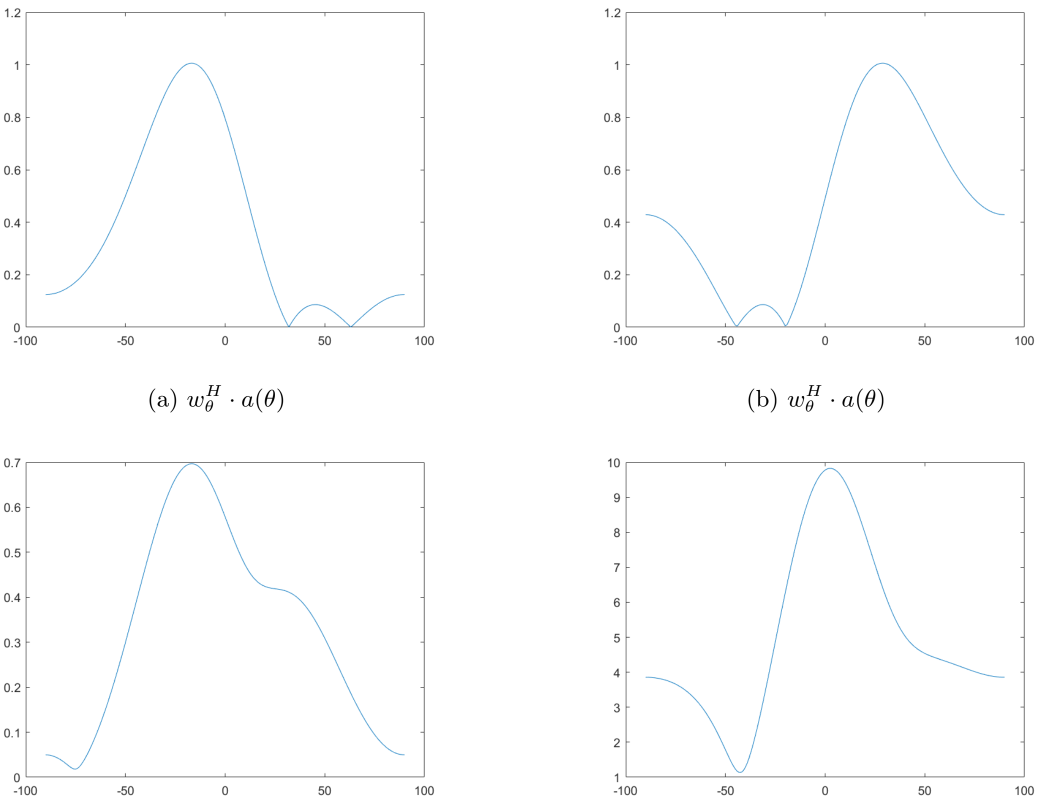
  $w^H = (A_f^H \cdot A_f)^{-1} \cdot A_f^H$

  return $S_f = w^H \cdot X$

end procedure

With both beamformers, the signal is perfectly recovered without the presence of noise.

5.3 Spatial Responses



(a) $w_\theta^H \cdot a(\theta)$

(b) $w_\theta^H \cdot a(\theta)$

(c) $w_f^H \cdot a(\theta)$

(d) $w_f^H \cdot a(\theta)$

Figure 4: $w^H \cdot a(\theta)$ estimation for angles in the range of $[-90, 90]$

Which gives the best suppression of interference? Why?

The beamformer based of the w^H built directly using θ is better at suppressing interferences. As one can see from the graphs, when (a) is provided a combined signal consisting of a source at $\theta = [-20, 30]$, only the signal with $\theta = -20$ is output, and vice-versa given (b) with the same input.

Compared to the beamformer built off the frequency estimation, one can see that, if the same combined signal is input, no signal is completely attentuated, and hence the interference is worse.

Looking at the pseudocode above, my first thought was that the angle estimation beamformer was better as we directly estimate the Array Response, while the frequency estimation beamformer first estimates the Source then the Array Response. However, looking at the generated signal, we can see that $\theta = [-20, 30]$ and $f = [0.1, 0.12]$, which means that the frequency difference is much lower than the angle difference, which I think is now what causes the frequency estimation beamformer to perform worse.

6 Channel Equalization - Signal Model

6.1 Task 1 - Construct X

Algorithm 7 Generate Data

```
procedure GENDATA_CONV(s, P, N, sigma)                    ▷ Inputs provided
    X = [2·P × N − 1]        ▷ It's N − 1 because we don't have the N + 1^th symbol
    for n = 1:N-1 do
        H = [P·2]
        for i = 1:P do
            H(i,1) = getH((i-1)/P)
            H(i+P,2) = getH((i-1)/P)
        end for
        X(:,n) = H * s(n:n+1)
        for i = 1:2·P do
            X(i,n) = X(i,n) + noise
        end for
    end for
    return X, H
end procedure
```

What is the rank of X and why?
The rank of cursive X is 2. Generally the first P rows of X are have the following values (I precised the first column due for simplicity): $(1 * s_0, -1 * s_0, 1 * s_0, -1 * s_0)^T$, which corresponds to a rank of 1. The second P rows of X have the following values: $(1 * s_1, -1 * s_1, 1 * s_1, -1 * s_1)^T$ which is also of rank 1. However as the first set and second set of rows are linearly independent (due to a different symbol used), the total rank of the matrix is 2. Generalizing this, I would say, for an $[mP \times (N-1)]$ matrix X, the rank of the matrix is m.

What if we double P? How does this change the rank of X and why?
As stated above, P does not affect the rank of the matrix X, m does. As such, doubling P does not change the rank of X. The situation is however difference if the noise standard deviation is not 0. In such as case the rank of the matrix is defined by mP, in which case, doubling P results in a doubling of the rank.

6.2 Zero-forcing and Wiener Receiver

Algorithm 8 Zero Forcing Receiver

```
procedure ZERO_FORCING(X, H)                               ▷ Inputs provided
    w^H = pinv(H)
    w^H = w^H(n, :)     ▷ Change n to any index to retrieve the nth symbol period
    S = w^H * X
    return S
end procedure
```

Algorithm 9 Wiener Receiver

---

procedure WIENER(X, H) ▷ Inputs provided

  $R_x = E(XX^H)$

  $R_{xs} = H$

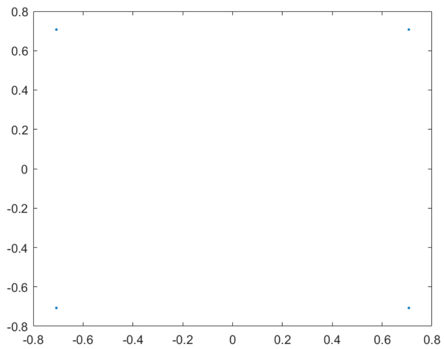
  $w = R_x^{-1} \cdot R_{xs}$

  $w^H = w^H(n, :)$ ▷ Change n to any index to retrieve the nth symbol period

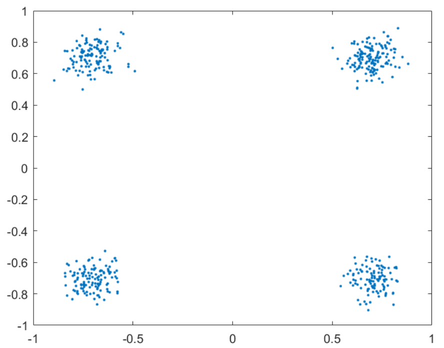
  S $= w^H * X$

  return S

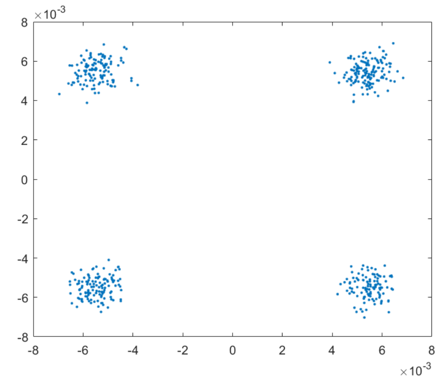
end procedure



(a) S



(b) Zero Forcing



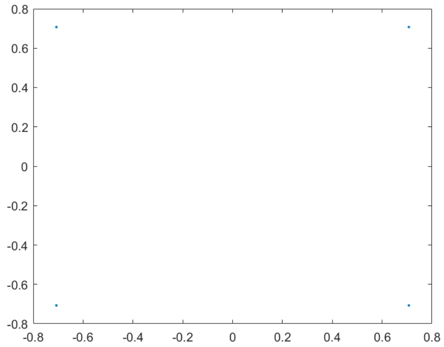
(c) Wiener

Figure 5: P = 4, delay 0

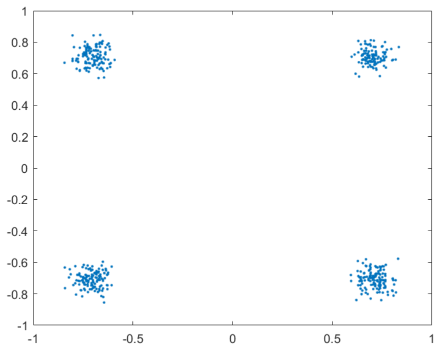
What is a good delay for these receivers?

In this section, I tally my observations of having played around with the two possible delays $0 and 1$. The first observation is that a delay of 0 allows the perfect recovery of my signal for the Zero-Forcing receiver. In the

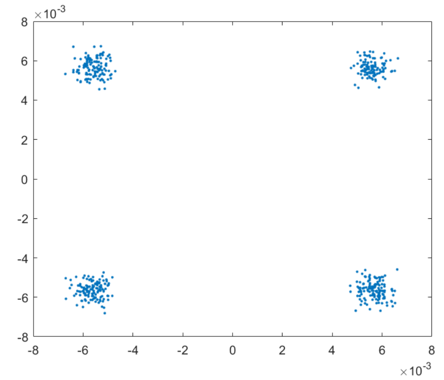
presence of noise, both for the ZF and Wiener receiver, using a delay of 0 allows for the estimated S to follow the same signs as the original signal. Using a delay of 1 results in a symbol sequence which I could not relate to the original signal.
However, plotting the clusters formed by both delays, they are very comparable.



(a) S



(b) Zero Forcing



(c) Wiener

Figure 6: P = 8, delay 0

Comparing this previous plot with the previous one, the immediate observation is the density of the clusters that are formed. By further oversampling, the original signal can more accurately be estimated.

7 Appendix

7.1 GenerateData

```
close all;
```

```
[X, A, S] = gendata(5, 20, 0.5, [20, 30], [0.20, 0.3], 20);

%plot(Singular,'*r')

%theta = esprit(X, 2);

%f = espritfreq(X, 2);

%[theta, f] = joint(X, 2, 10);

%[U,S,V] = svd(X,"econ");
%plot(diag(S), 'o');





esp_theta = zeros(2, 6);
sd_theta = zeros(2, 6);
esp_f = zeros(2, 6);
sd_f = zeros(2, 6);
jt_theta = zeros(2, 6);
sd_jd_theta = zeros(2, 6);
jt_f = zeros(2, 6);
sd_jd_f = zeros(2, 6);

for snr = 0:4:20
    esprit_theta = zeros(2, 1000);
    esprit_f = zeros(2, 1000);
    joint_theta = zeros(2, 1000);
    joint_f = zeros(2,1000);

    for i = 1:1000
        [X, A, S] = gendata(3, 20, 0.5, [-20, 30], [0.1, 0.12], snr);

        theta = esprit(X, 2);
        theta = sort(theta);
        esprit_theta(1,i) = theta(1);
        esprit_theta(2,i) = theta(2);

        f = espritfreq(X, 2);
        f = sort(f);
        esprit_f(1,i) = f(1);
        esprit_f(2,i) = f(2);

        [theta, f] = joint(X, 2, 10);
        theta = sort(theta);
        f = sort(f);
        joint_theta(1,i) = theta(1);
        joint_theta(2,i) = theta(2);

        joint_f(1,i) = f(1);
        joint_f(2,i) = f(2);

    end
```

```
        esp_theta(:,snr/4+1) = mean(esprit_theta, 2);
        sd_theta(:,snr/4+1) = std(esprit_theta, 0, 2);

        esp_f(:,snr/4+1) = mean(esprit_f, 2);
        sd_f(:,snr/4+1) = std(esprit_f, 0, 2);

        jt_theta(:,snr/4+1) = mean(joint_theta, 2);
        sd_jd_theta(:,snr/4+1) = std(joint_theta, 0, 2);

        jt_f(:,snr/4+1) = mean(joint_f, 2);
        sd_jd_f(:,snr/4+1) = std(joint_f, 0, 2);

end

f1 = figure('Name','ESPRIT THETA');
hold on
plot((0:4:20), sd_theta(1,:),'r.');
plot((0:4:20), sd_theta(2,:),'ro');
% Add joint
hold off

f2 = figure('Name','ESPRIT F');
hold on
plot((0:4:20), sd_f(1,:),'g.');
plot((0:4:20), sd_f(2,:),'go');
% Add joint
hold off

f3 = figure('Name','JOINT THETA');
hold on
plot((0:4:20), sd_jd_theta(1,:),'g.');
plot((0:4:20), sd_jd_theta(2,:),'go');
% Add joint
hold off

f4 = figure('Name','JOINT F');
hold on
plot((0:4:20), sd_jd_f(1,:),'g.');
plot((0:4:20), sd_jd_f(2,:),'go');
% Add joint
hold off

% [X, A, S] = gendata(3, 20, 0.5, [-20, 30], [0.1, 0.12], 10);
%
% theta = esprit(X, 2);
% f = espritfreq(X, 2);
%
% [S_theta, w_H_theta] = zero_forcing_theta(X, 0.5, theta);
% [S_f, w_H_f] = zero_forcing_freq(X, f);
%
% plot_spatial_response_theta(w_H_theta, 0.5); %TODO: Add magnitude
%
% plot_spatial_response_f(w_H_f, 0.5); %TODO: Add magnitude

function [X, A, S] = gendata(M, N, Delta, theta, f, SNR)
    % Create empty matrix MxN -> ReceiverAntenna x SamplesMeasured
    X = zeros(M, N);

```

```
    % S = source vector
    % Create and initialize vector of sources (irrespective of receiver
    % antenna)
    num_sources = size(f, 2);
    S = zeros(num_sources, 1);
    for i = 1:num_sources
        S(i) = exp(1i*2*pi*f(i));
    end

    % NOTE: x(t) = A*s(t) + n(t)
    % A = attenuation caused by angle and antenna distance (delta)
    % k = M?
    A = zeros(M, num_sources);
    for i = 0:M-1
        for j = 1:num_sources
            A(i+1,j) = exp(1i*2*pi*i*Delta*sind(theta(j)));
        end
    end

    % Add Noise - AWGN
    for i = 1:N
        S_prime = S.^i;
        temp = A * S_prime;
        X(:,i) = awgn(temp, SNR);
    end
end

function theta = esprit(X, d)
    X_top = X(1:size(X,1) - 1, :);
    X_bottom = X(2:size(X,1), :);

    Z = [X_top; X_bottom];

    [U,S,V] = svd(Z,"econ");

    U = U(:,1:d);

    %Instead of this for loop to cut down small values, I think d needs to
    %be used since we assume the number of sources
    %for i = size(S,1):-1:1
    %    if S(i,i) < 0.00005
    %        U(:,i) = [];
    %        S(i,:) = [];
    %        S(:,i) = [];
    %    end
    %end

    Ux = U(1:size(U,1)/2, :);
    Uy = U(size(U,1)/2+1:size(U,1), :);

    pUx = pinv(Ux);
    pUxUy = pUx * Uy;

    [Vectors,Values] = eig(pUxUy);

    theta = zeros(size(Values,1),1);

    for i = 1:size(Values,1)
```

```
            theta(i) = 180/pi*asin(angle(Values(i,i))/pi);
        end
end

function f = espritfreq(X, d)
        x_t = X(1,:);

        N = size(x_t, 2);
        m = N/2;

        Z = zeros(m, N-m);

        for j = 1:N-m
            Z(:,j) = x_t(1+(j-1):m+(j-1));
        end

        [U,S,V] = svd(Z,"econ");

        U = U(:,1:d);

        Ux = U(1:size(U,1) - 1, :);
        Uy = U(2:size(U,1), :);

        pUx = pinv(Ux);
        pUxUy = pUx * Uy;

        [Vectors,Values] = eig(pUxUy);

        f = zeros(size(Values,1),1);

        for i = 1:size(Values,1)
            f(i) = angle(Values(i,i)) / (2*pi);
        end
end

function [theta, f] = joint(X, d, m)
        N = size(X, 2);

        Z = zeros(m*size(X, 1), N-m);

        for j = 1:N-m
            counter = 0;
            for i = 1:size(X, 1):m*size(X, 1)
                Z(i:i+size(X, 1)-1,j) = X(:,j+counter);
                counter = counter + 1;
            end
        end

        [U,S,V] = svd(Z,"econ");

        U = U(:,1:d);

        deltaX = [eye(size(X,1)) zeros(size(X,1),1)];
        deltaY = [zeros(size(X,1),1) eye(size(X,1))];

        tempX = transpose(kron(eye(m), deltaX));
        tempY = transpose(kron(eye(m), deltaY));

```

```
    Ux = tempX * U;
    Uy = tempY * U;

    pUx = pinv(Ux);
    pUxUy_theta = pUx * Uy;

%     [Vectors,Values] = eig(pUxUy_theta);
%     theta = zeros(size(Values,1),1);
%     for i = 1:size(Values,1)
%         theta(i) = -180/pi*asin(angle(Values(i,i))/pi);
%     end

    deltaX = transpose(kron([eye(m) zeros(m,1)], eye(size(X,1))));
    deltaY = transpose(kron([zeros(m,1) eye(m)], eye(size(X,1))));

    Ux = deltaX * U;
    Uy = deltaY * U;

    pUx = pinv(Ux);
    pUxUy_f = pUx * Uy;

    % Solving Joint Diagonalization
    M = [pUxUy_theta pUxUy_f];
    [V,D] =  joint_diag(M,1e-8);

    D1 = D(:,1:d);
    D2 = D(:,d+1:2*d);

    theta_tmp=diag(D1);
    theta = -asin(angle(theta_tmp)./(pi))*180/pi;

    phi = diag(D2);
    f=-angle(phi)/(2*pi);


    [theta, index] = sort(theta);

    f = f(index);

%     [Vectors,Values] = eig(pUxUy_f);
%     f = zeros(size(Values,1),1);
%     for i = 1:size(Values,1)
%         f(i) = -angle(Values(i,i)) / (2*pi);
%     end
end

function [S_estimate, w_H] = zero_forcing_theta(X, Delta, theta)
    M = size(X, 1);
    num_sources = size(theta,1);

    A = zeros(M, num_sources);
    for i = 0:M-1
        for j = 1:num_sources
            A(i+1,j) = exp(1i*2*pi*i*Delta*sind(theta(j)));
        end
    end

    tempa = A';
```

```
    tempb = A' * A;
    tempc = eye(num_sources)/tempb;

    w_H = tempc * tempa;

    S_estimate = w_H * X;
end

function [S_estimate, w_H] = zero_forcing_freq(X, f)
    num_sources = size(f, 1);
    num_samples = size(X, 2);

    S = zeros(num_sources, num_samples);
    for i = 1:num_sources
        for j = 1:num_samples
            S(i,j) = exp(1i*2*pi*f(i)*j);
        end
    end

    tempa = S';
    tempb = S*tempa;
    tempc = eye(num_sources) / tempb;
    A = (X * tempa) * tempc;

    tempa = A';
    tempb = A' * A;
    tempc = eye(num_sources)/tempb;

    w_H = tempc * tempa;
    S_estimate = w_H * X;
end

function spatial_responses = plot_spatial_response_theta(w_H, Delta)
    M = size(w_H, 2);
    num_sources = size(w_H, 1);

    y = zeros(2,180);

    for angle = -90:90
        A = zeros(M, num_sources);
        for i = 0:M-1
            for j = 1:num_sources
                A(i+1,j) = exp(1i*2*pi*i*Delta*sind(angle));
            end
        end
        temp = w_H * A;
        y(1,angle+91) = abs(temp(1,1));
        y(2,angle+91) = abs(temp(2,1));
    end

    x_axis = (-90:90);
    f3 = figure;
    plot(x_axis, y(1,:));

    f4 = figure;
    plot(x_axis, y(2,:));
end
```

```
function spatial_responses = plot_spatial_response_f(w_H, Delta)
    M = size(w_H, 2);
    num_sources = size(w_H, 1);

    y = zeros(2,180);

    for angle = -90:90
        A = zeros(M, num_sources);
        for i = 0:M-1
            for j = 1:num_sources
                A(i+1,j) = exp(1i*2*pi*i*Delta*sind(angle));
            end
        end
        temp = w_H * A;
        y(1,angle+91) = abs(temp(1,1));
        y(2,angle+91) = abs(temp(2,1));
    end

    x_axis = (-90:90);
    f7 = figure;
    plot(x_axis, y(1,:));

    f8 = figure;
    plot(x_axis, y(2,:));
end
```

7.2 ChannelEqualization

```
close all;
temp = 1/sqrt(2) + 1i*1/sqrt(2);
s = zeros(500, 1);
for i = 1:500
    a = rand(1);
    b = rand(1);

    if a >= 0.5 && b >= 0.5
        s(i) = 1/sqrt(2) + 1i*1/sqrt(2);
    elseif a >= 0.5 && b < 0.5
        s(i) = 1/sqrt(2) - 1i*1/sqrt(2);
    elseif a < 0.5 && b >= 0.5
        s(i) = -1/sqrt(2) + 1i*1/sqrt(2);
    else
        s(i) = -1/sqrt(2) - 1i*1/sqrt(2);
    end
end

%s(1) = 0 + 1i * 0;

[X, H] = gendata_conv(s, 4, 500, 0.5);
%X2 = gendata_conv2(s, 4, 500, 0);

% Zero Forcing Receiver
wH_ZF = pinv(H);
wH_ZF = wH_ZF(1,:);
ZF_S = wH_ZF * X;

% Wiener Receiver
E = (1/size(X,1))*(X * X');
Rx = sum(E, 'all');
```

```
Rxs = H;
w_W = Rxs / Rx;
wH_W = w_W';
wH_W = wH_W(1,:);
W_S = wH_W * X;

% Plotting
f1 = figure();
plot(ZF_S, '.');

f2 = figure();
plot(W_S, '.');

f3 = figure();
plot(s, '.')


function [X, H] = gendata_conv(s, P, N, sigma)

    X = zeros(2*P, N-1);
    for n = 1:N-1
        H = zeros(P, 2);

        for i = 1:P
            H(i,1) = getH(0 + (i-1)/P);
            H(i+P,2) = getH(0 + (i-1)/P);
        end

        X(:,n) = H * s(n:n+1);

        for i = 1:2*P
            X(i,n) = X(i,n) + (sigma.*rand(1, 1) + 1i*sigma.*rand(1,1));
        end
    end
end

function x = gendata_conv2(s, P, N, sigma)
    x = zeros(2*P, N);

    for i = 1:2*P
        for j = 1:N
            x(i,j) = sigma.*rand(1, 1) + 1i*sigma.*rand(1,1);
        end
    end
    w = x;

    for i = 1:2*P
        for j = 1:N
            h = zeros(N,1);
            for k = 1:N
                h(k) = getH((j-1) - (k-1) + (i-1)/P);
            end

            conv_res = conv(s, h, 'valid');
            x(i,j) = x(i,j) + conv_res;
        end
    end
end
```

```
function h = getH(t)
    if t >= 0 && t < 0.25
        h = 1;
    elseif t >= 0.25 && t < 0.5
        h = -1;
    elseif t >= 0.5 && t < 0.75
        h = 1;
    elseif t >= 0.75 && t < 1
        h = -1;
    else
        h = 0;
    end
end
```