Optimization of Beamforming

Zichao Zhang

Project report of ELG 6108
Introduction to Convex Optimization

Department of Systems and Computer Engineering
Carleton University
March 14, 2022

Contents

Abstract

Linear sensor arrays or antenna arrays had been studied and applied in practice extensively since last century, beamforming is the enabler technology behind it. It's a technology that boost the spectral efficiency and capacity at look direction, and if designed properly, can also eliminate interference at specific direction. The most popular way to design a beamformer is through optimization, especially convex optimization, which is a subject studied and used extensively and there's also plenty of efficient software packages available. In this report, we will explore several popular papers that introduces ways to design an optimal beamformer under certain conditions.

1 Contributions

I did simulation of [4], [2], [1], compared the performace of last two in same figure, explored the effect of learning rate to the convergence in Frost's algorithm. In the simulation of second paper by Frost, he only talked about signal come from broadside, namely, 90°, I simulated the case signal come from an azimuth, not 90°.

2 introduction

In recent time, beamforming has become a key enabler for 5G and even beyond 5G technologies, concepts like mmWave communication and massive MIMO requires performing beamforming to function properly. Beamforming is applied in wi-fi router and LTE standard to boost capacity. Traditional single antenna transmitter radiates electromagnetic wave isotropicly, and transmit power is also uniformly distributed at wavefront, which means antenna radiates power at every direction. This is meaningful in a sense that if we need broadcasting, isotropic transmitter is preferable. However, spending energy and power to cover those directions that will never or seldomly require coverage is a waste of resource. In late 1940s, people had been using beamforming to improve sonar during WWII, in fact anyform of energy travels in wave can use beamforming to improve its performance.

The idea of beamforming is to deploy a linear array of antennas (also called uniform linear array, ULA) and use antenna pattern brought by phase difference from different arrive time to form a beam towards the direction we want. There are also other types of array arrangements like circular, but we won't talk about it here. Beamforming focuses radiation power toward one direction, and gain of other directions are reduced. This property of beamforming saves power from radiating them everywhere and also boosts spectral efficiency of indicated direction. Beammforming requires signal processing technologies at transmission and reception, according to antenna theory, we need complex beamforming vector to match the upcoming signal. Sensor array (antenna array) is first used in sonar and radar, which means it should be able to detect objects on the move, it is also the case in wireless mobile communication, beamforming should form beam towards moving users. In any physical systems, we will face noise generated by electronic circuits or from nature, there will also be interference from other users or from other base stations. This requires us to design a receive filter (receive beamformer) that is pointing the beam towards designated direction as well as suppressing all the noise and interference.

In this report, I will have a literature review on several outstanding research papers about adaptive array processing. Then I will present the methods used in those papers and show the simulation results of their method.

3 Literature Review

3.1 Sample Matrix Inversion (SMI) Algorithm

The first paper by Reed [4] introduces an adaptive algorithm that converges fast. In practical detection, signal strength can be relatively lower compared to interference, then it is extremely important to supress the effect of interference. According to Reed, to detect the signal, we need to maximize SNR, where SNR in the paper is actually SINR, signal to interference plus onise ratio. The SINR has formulation of non-convex form, but he didn't use convex optimization tools to find solution, rather, simply applied Cauchy-Schwartz inequality and got optimum. The optimal solution requires the inverse of covariance matrix of interference and noise, which is not known to designers, and this brings his algorithm, sample matrix inversion (SMI). The idea of SMI algorithm is to have an estimation of real covariance matrix, the receiver samples the channel, or receive a sequence of samples, then use them to compute sample covariance matrix, after which we can compute the inverse and use it to form our beamformer.

One concern is that the sample matrix is an estimation of interference and noise matrix, to compute it we require samples of noise and interference. This tells us we need to sample the channel for a period of time to form beamformer to supress noise and no presence of signal is allowed. That requirement is achieviable under radar detection but becomes impractical in the context of communication. In modern cellular communication, receiver receives interference from many changing directions, and interference is not constant because of fading, then we need the estimation time short enough to deal with interference timely. Besides, we also have to deal with the presence of signal because abscene of signal means break of communication. As indicated in the paper, the algorithm performance decays with signal present.

The algorithm only requires small number of samples, and it converges fast. But inversion of a matrix makes it computationally costing. In the process of optimization, there isn't any constraint on noise gain nor signal gain, so we're maximizing SNR but we might also lose control over noise supression.

3.2 Constrained Least Mean Square (LMS) Algorithm

Frost [2] proposed an adaptive algorithm that iteratively updates receiver beamformer, or weights. The problem is convex and has linear constrains, so it is called constrained least mean squares (LMS) problem. He utilized stochastic gradient descent to update weights while keeping constraint satisfied at each step. The model for the problem is at the receiver there's presence of both signal and noise plus interference, Frost aimed to minimize the total output power while maintaining the gain at look direction, it is equivalent to maximize SINR. The linear constraints span a subspace where the optimal solution must be. The algorithm has multiple linear constraints which keep gain at look direction unchanged. The optimization problem can be solved efficiently by Lagrange multiplier, but there's still inversion of covariance matrix of joint signal and noise. As we don't know the true distribution of signal or noise, we can still use sample covariance matrix to estimate it.

But Frost applied gradient descent method to iteratively update the weights, updating vector moves to the negative direction of gradient at each iteration, instead of using real covariance matrix, sample at receiver is used, signal and noise sample jointly. But there's no guarantee that at each update, vector will still satisify constraint. Therefore, a projection matrix is used at each iteration to project updating vector onto the constraint space so that gain at signal direction is unchanged.

The algorithm only requires a few multiplication and addition at each step, because only one sample is required and no need for matrix inversion. With projection matrix, round-off errors will not accumulate so the algorithm has error correcting property. According to the simulation results, the algorithm requires at least 30 samples to present significant convergence. Therefore, compared to SMI algorithm, this algorithm may require longer convergence time, but constrained LMS doesn't need to break communication and sample the noise, the updating procedure can proceed during the communication.

3.3 Improved Recursive Adaptive Beamforming

In [1], Cox improved the algorithm of Frost. In his paper, it is assumed there will be errors occur in real or practical systems, an optimal beamformer should be robust to those errors, namely, insensitive to errors. Small phase, amplitude or position errors will lead to signal suppression effect, but in physical systems, errors in each sensor are usually uncorrelated to each other, so they are modeled as spatially white noise in the paper. So gain against white noise becomes a criterion of robustness. In the paper this gain is defined as white noise gain and used as quadratic constraint for the convex problem.

The improved algorithm tries to minimize output power while keeping linear constraints of Frost's algorithm, moreover, another quadratic constraint is added as constraint on white noise gain. Updating procedure is still the same as Frost's algorithm but he chose to update the part that is orthogonal to the subspace spanned by linear constraints, because the updating vector can be decomposed into part orthogonal to that space and part in that space. In each iteration, the vector still satisfies linear constraint because projection onto subspace is untouched. And when updating weights, we still need to satisify quadratic constraint, the quadratic constraint can be transformed to circular constraint, in other words, the constraint becomes constraint on the norm. Therefore, in each step, if the norm exceeds constraint, we need to normalize it to guarantee white noise gain. Similarly, in each step constraints are satisfied so that round off error won't accumulate.

3.4 Adaptive beamforming robust to signal mismatch problem

In previous introduced algorithms, signal mismatch will usually decay the performance dramatically. Vorobyov [5] proposed an algorithm robust to signal mismatch. The problem formulation is similiar to other problems, convex problems with constraints. In order to overcome signal supression caused by signal mismatch, deviated signal is regulated to be around the broadside, all the possible deviated signal vectors compose a set, gain on any element in the set has to be greater than a threshold. But each constraint here belongs to the set and there're infinite constraints, this makes the problem non-convex.

However, the semi-infinite nonconvex constraint can be replaced by the worst-case construction, and constraint becomes single convex constraint, the problem thereby becomes convex. The solution of the worst case problem can be solved by Lagrange multiplier but there's no closed form solution. The problem is further transformed into a second order cone (SOC) problem, and it's solved efficiently by interior point algorithm, using software package. The computation is comparable to SMI algorithm so it require powerful hardware.

3.5 User Grouping in HAPS communication

As the highly demand for flexible resource allocation and robust performance for massive users, there's a need for mobility of base stations, there comes high altitude platform stations (HAPS). HAPS is a promising solution for next generation network in urban area, it has dependency on LOS component and therefore more suitable to mount technologies like massive MIMO and mmWave. In [3], they proposed a user grouping scheme that uses statistical eigenmode to design outer beamformer.

The optimization problem is convex, but also transformed into linear problem (LP). Based on the optimization, an iterative algorithm is proposed, it is based on the chordal distance between statistical eigenmodes of users in the group, and then according to which to divide users. The paper demonstrates signal power is mainly in statisticcal eigenmodes, their method used reduced dimensional of the statistical eigenmode.

4 Technical Details

4.1 Notation

In this report, vectors are represented by lowercase bold font, and randon vectors are denoted as bold capital italic and random variables are represented as capital italic, matrices are represented by bold capital. Superscript on symbol a^+ means Hermitian transpose, and a^* means complex conjugate while a^T means transpose. $\mathbb{E}$ represents expectation.

4.2 System model

In radar and sonar, it's a slight different story with communication, sensor array has M elements, but the sensor need to sample L times, then we have our system model. The sensor array is composed of M elements as discussed, and samples L times, which makes a sampled data vector $\boldsymbol{X} \in \mathbb{R}^{ML\times 1}$, let $N = ML$. The sampled data is composed of three (or two) components, signal, interferece and noise. Represent interference and noise as $\boldsymbol{N}$, and signal as $\boldsymbol{S}$, both of them are $N \times 1$ vectors, we have

$$\boldsymbol{X} = \boldsymbol{S} + \boldsymbol{N} \tag{1}$$

In communication subject, direct component carries no information, so in practice people would avoid direct component in the signal. Thus, it is preferable to have $\mathbb{E}[\boldsymbol{S}] = \mathbf{0}$. The noise

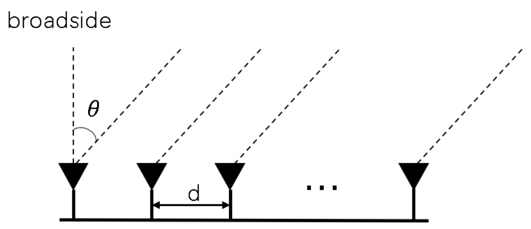


Figure 1: Array of antennas

comes from interference caused by other users, which is also signal used for communication with zero mean, and white noise produced by electronic devices, usually modeled as zero mean white Gaussian noise. Thus, we have $\mathbb{E}[\boldsymbol{X}] = \mathbf{0}$.

4.2.1 Antenna pattern

In electromagnetic theory, a plane wave arrives at an linear array of antennas with order, namely, different antenna elements recives signal in different time, and this causes phase difference in the received component and causes attenuation in some cases. Assume we have a linear array and distance between two elements is d, every element is an isotropic antenna, signal comes at an angle θ with broadside, we have a model like Fig. 1.

There will be phse differences between elements, wavefront arrives at first antenna, after traveling $\Delta l = d \sin\theta$, wavefront arrives the second element. Therefore, this difference in propagation distance results in a difference in phase, and this difference between two antennas is the same. We assume radian frequency is ω, to travel Δl, signal needs time $\Delta t = \Delta l / c$, where c is speed of light, then phse difference $\Delta\phi$:

$$\Delta\phi = \omega\Delta t = \frac{2\pi f_c \Delta l}{c} = 2\pi\frac{d}{\lambda}\sin\theta \tag{2}$$

where λ is wave length and f_c is carrier frequency, after receiving the signal, receiver will multiply each entry with a constant, that is called receiver filter, and this operation is called beam forming. We call the receive filter $\boldsymbol{w}$, and structure of receiver is shown as Fig. 2.

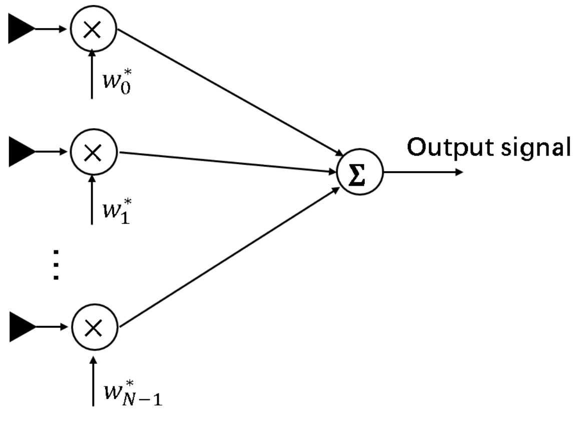


Figure 2: Beamforming model

Received signal is $\boldsymbol{X}$, We can also represent beamforming as

$$Y = \sum_{i=1}^{N} w^* X_i = \boldsymbol{w}^+ \boldsymbol{X} \tag{3}$$

Assume our filter $\boldsymbol{w} = [1, 1, \cdots, 1]^T$, and a fixed signal $x \times [1, 1, \cdots, 1]^T$, the received signal is

$$y = x \sum_{i=1}^{N} e^{j(i-1)\Delta\phi} = x \sum_{i=1}^{N} e^{j2\pi(i-1)\frac{d}{\lambda}\sin\theta} \tag{4}$$

Therefore, we have antenna pattern or ULA pattern

$$F(\theta) = \frac{|\sum_{i=1}^{N} e^{j(i-1)\Delta\phi}|}{max\{|\sum_{i=1}^{N} e^{j(i-1)\Delta\phi}|\}} = \left|\frac{\sin(n\pi\frac{d}{\lambda}\sin\theta)}{n\sin(\pi\frac{d}{\lambda}\sin\theta)}\right| \tag{5}$$

It can be seen in the following figure

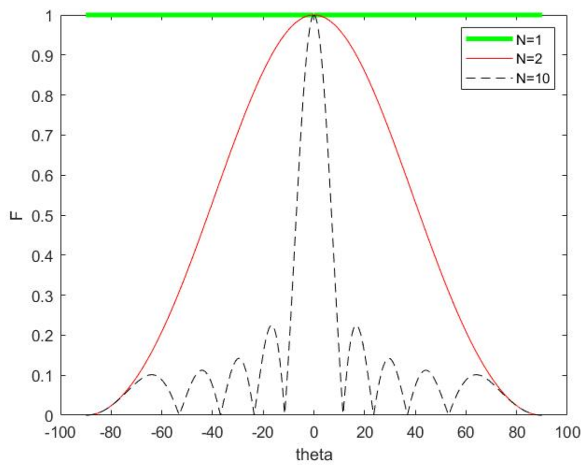


Figure 3: Antenna pattern with $d = \frac{\lambda}{2}$, and shown with $n = 1, 2, 10$

If we change $\boldsymbol{w}$, we can control this pattern and change the maximum response angle to where we need, this is called beam steering, this means, if we want the transmit or receive beam to point towards θ_0, we can let $w_i = e^{j(i-1)2\pi\frac{d}{\lambda}\sin\theta_0}$, and this is shown below

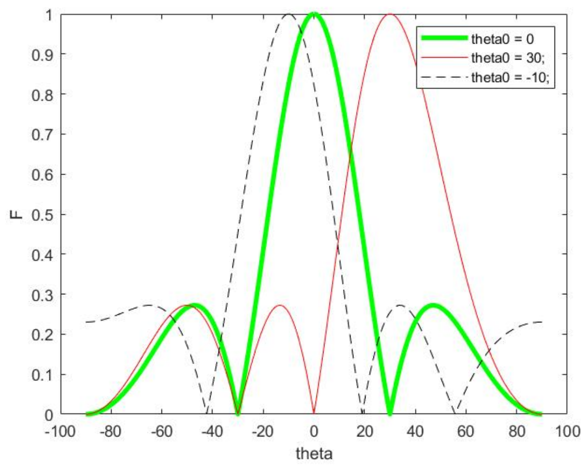


Figure 4: Antenna pattern with beam steering, $d = \frac{\lambda}{2}$, and shown with $\theta_0 = 0, 30, -10$

4.3 SMI Algorithm

From (3), we know that mean value of output $\mathbb{E}[Y] = 0$, and $Var[Y] = \boldsymbol{w}^{+}\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}$, where

$$\mathbf{R}_{\boldsymbol{X}} = \mathbb{E}\big[\boldsymbol{X}\boldsymbol{X}^{+}\big] = \boldsymbol{S}\boldsymbol{S}^{+} + \mathbb{E}\big[\boldsymbol{N}\boldsymbol{N}^{+}\big] = \boldsymbol{S}\boldsymbol{S}^{+} + \mathbf{R}_{\boldsymbol{N}} \tag{6}$$

Covariance matrix of noise (noise plus interference) is usually full rank, because

$$\mathbf{R}_{\boldsymbol{N}} = \frac{N_0}{2}\mathbf{I} + \mathbf{R}_{\boldsymbol{I}} \tag{7}$$

where $\mathbf{R}_{\boldsymbol{I}}$ is the covariance matrix of interference, if noise and interference are uncorrelated. $\mathbf{R}_{\boldsymbol{N}}$ is positive definite and symmetric, then it can be diagonalized, we can find unitary matrix $\mathbf{U}$ that

$$\mathbf{U}^{+}\mathbf{R}_{\boldsymbol{N}}\mathbf{U} = \boldsymbol{\Lambda} = diag(\lambda_1, \lambda_2, \cdots, \lambda_N) \tag{8}$$

$\lambda_1, \lambda_2, \cdots, \lambda_N$ are non-negative eigenvalues of covariance matrix of noise, in the paper, λ is used to represent eigenvalues of $\mathbf{R}_{\boldsymbol{I}}$, but $\mathbf{R}_{\boldsymbol{I}}$ is not used in later discussions, we focus on $\mathbf{R}_{\boldsymbol{N}}$ for now.

The output of receiver has two parts, one is signal part, represented by $\boldsymbol{Y}_s = \boldsymbol{w}^{+}\boldsymbol{S}$, the other is noise part, represented by $\boldsymbol{Y}_n = \boldsymbol{w}^{+}\boldsymbol{N}$, so output SNR is

$$SNR = \frac{\mathbb{E}\big[\big|\boldsymbol{Y}_s\big|^2\big]}{\mathbb{E}\big[\big|\boldsymbol{Y}_n\big|^2\big]} = \frac{\mathbb{E}\big[\big|\boldsymbol{w}^{+}\boldsymbol{S}\big|^2\big]}{\mathbb{E}\big[\big|\boldsymbol{w}^{+}\boldsymbol{N}\big|^2\big]} = \frac{\big|\boldsymbol{w}^{+}\boldsymbol{S}\big|^2}{\boldsymbol{w}^{+}\mathbf{R}_{\boldsymbol{N}}\boldsymbol{w}} \tag{9}$$

so our problem is to maximize SNR, but notice that there's no constraint, and this problem is not convex, but constraint is not required to solve the problem, convexity also doesn't matter here, we simply let $\boldsymbol{v} = \mathbf{R}_{\boldsymbol{N}}^{\frac{1}{2}}\boldsymbol{w}$ and $\boldsymbol{w} = \mathbf{R}_{\boldsymbol{N}}^{-\frac{1}{2}}\boldsymbol{v}$ plug in the original expression and apply Cauchy-Schwartz inequality, we get

$$SNR = \frac{\big|\boldsymbol{v}^{+}\mathbf{R}_{\boldsymbol{N}}^{-\frac{1}{2}}\boldsymbol{S}\big|^2}{\big|\boldsymbol{v}\big|^2} \leq \frac{\big|\boldsymbol{v}\big|^2\big|\mathbf{R}_{\boldsymbol{N}}^{-\frac{1}{2}}\boldsymbol{S}\big|^2}{\big|\boldsymbol{v}\big|^2} = \big|\mathbf{R}_{\boldsymbol{N}}^{-\frac{1}{2}}\boldsymbol{S}\big|^2 \tag{10}$$

the equality holds iff $\boldsymbol{v} = k\mathbf{R}_{\boldsymbol{N}}^{-\frac{1}{2}}\boldsymbol{S}$, k is some constant. Then, $\boldsymbol{w}^* = k\mathbf{R}_{\boldsymbol{N}}^{-1}\boldsymbol{S}$, $\boldsymbol{w}^*$ is the optimal beamformer, then SNR becomes

$$SNR = \frac{\left|k\boldsymbol{S}^+\boldsymbol{R}_{\boldsymbol{N}}^{-1}\boldsymbol{S}\right|^2}{\left|k\right|^2\boldsymbol{S}^+\boldsymbol{R}_{\boldsymbol{N}}^{-1}\boldsymbol{S}} = \boldsymbol{S}^+\boldsymbol{R}_{\boldsymbol{N}}^{-1}\boldsymbol{S} \tag{11}$$

In practice, one does not know this covariance matrix, or true distribution, then in order to make an estimation for this matrix, we use sample covariance matrix instead. As we know, expectation operation can be replaced by average, so our estimated matrix is

$$\hat{\mathbf{R}}_{\boldsymbol{N}} = \frac{1}{K}\sum_{i=1}^{K}\boldsymbol{n}_i\boldsymbol{n}_i^+ = \frac{1}{K}\mathbf{N}^+\mathbf{N} \tag{12}$$

where $\hat{\mathbf{R}}_{\boldsymbol{N}}$ is our estimation of real covariance matrix, $\boldsymbol{n}_i$ is the ith noise sample, from here we can see that this algorithm requires abscene of signal, and $\mathbf{N}$ is $K \times N$ sample matrix with ith row $\boldsymbol{n}_i^T$.

4.4 Constrained LMS Algorithm

The notation used by Frost is a little different with those used in this report, so we adapt them to notations used in this report to keep continuity. First of all, we need to have linear constraints to keep look direction gain a constant. The constraint is represented as matrix product,

$$\mathfrak{F} = \mathbf{C}^+\boldsymbol{w} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_L]^+\boldsymbol{w} \tag{13}$$

where $\boldsymbol{c}_i's$ are linear constraints, the $N \times 1$ vectors are composed of L group of entries, with each group M entries and $\boldsymbol{c}_i = [0, 0, \cdots, 1, 1, 1, \cdots, 0, 0]^T$, namely, $\boldsymbol{c}_i$ only has one at ith group. And $\mathfrak{F}$ is our gain at look direction, $\mathfrak{F} = [f_1, f_2, \cdots, f_L]^T$. Thus, with constraints, we can formulate the problem

$$\underset{\boldsymbol{w}}{minimize} \quad \boldsymbol{w}^+\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w} \tag{14}$$

$$\text{subject to } \mathbf{C}^+\boldsymbol{w} = \mathfrak{F} \tag{15}$$

The idea is minimize total output power while maintaining signal component power, then power of noise and interference is minimized, we will achieve noise suppression. This is a classical form of convex problem, therefore we can use Lagrange multiplier to find its solution, form Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}) = \boldsymbol{w}^+\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w} + \boldsymbol{\lambda}^T(\mathbf{C}^+\boldsymbol{w} - \mathfrak{F}) \tag{16}$$

From stationarty condition, we let the gradient of Lagrangian to be 0, and from primal feasibility, we have

$$\nabla\mathcal{L} = \mathbf{R}_{\boldsymbol{X}}\boldsymbol{w} + \mathbf{C}\boldsymbol{\lambda} = 0 \tag{17}$$

$$\boldsymbol{w}^* = -\mathbf{R}_{\boldsymbol{X}}^{-1}\mathbf{C}\boldsymbol{\lambda} \tag{18}$$

$$\mathbf{C}^+\boldsymbol{w} = \mathfrak{F} = -\mathbf{C}^T\mathbf{R}_{\boldsymbol{X}}^{-1}\mathbf{C}\boldsymbol{\lambda} \tag{19}$$

$$\rightarrow\boldsymbol{\lambda} = -[\mathbf{C}^T\mathbf{R}_{\boldsymbol{X}}^{-1}\mathbf{C}]^{-1}\mathfrak{F} \tag{20}$$

Plug in $\boldsymbol{\lambda}$, we get

$$\boldsymbol{w}^* = \mathbf{R}_{\boldsymbol{X}}^{-1}\mathbf{C}[\mathbf{C}^T\mathbf{R}_{\boldsymbol{X}}^{-1}\mathbf{C}]^{-1}\mathfrak{F} \tag{21}$$

According to Frost [2], by proper choice of $\mathfrak{F}$, or properly choose response, we can get a variety of optimum beamformers. A concern is that we still have matrix inversion and this time we even have three matrix inversions, that's triple the amount of computation of SMI method, so there's a need for adaptive algorithm.

We still assume all the correlation matrices are unknown, and we get the optimum weight (or beamformer) by iteratively update, that is the gradient descent constrained LMS algorithm. We start with a weight that satisfies constraint, $\boldsymbol{w}(0) = \mathbf{C}[\mathbf{C}^T\mathbf{C}]^{-1}\mathfrak{F}$, for each iteration step, a updating vector is added to $\boldsymbol{w}$, namely, weight moves to negative gradient direction, which indicates the fastest descent direction. Like what we did in machine learning, the learning rate controls how fast the algorithm descends, so we scale the gradient at each step by μ, the length of step is constrained in case of gradient explode. The weight at $k+1$ should be computed by

$$\begin{aligned}\boldsymbol{w}(k+1) &= \boldsymbol{w}(k) - \mu\nabla\mathcal{L}[\boldsymbol{w}(k)]\\ &= \boldsymbol{w}(k) - \mu[\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}(k) + \mathbf{C}\boldsymbol{\lambda}(k)]\end{aligned} \tag{22}$$

In this case, Lagrange multipliers are chosen to let $\boldsymbol{w}(k+1)$ satisify constraint at each step,

$$\mathfrak{F} = \mathbf{C}^+\boldsymbol{w}(k+1) = \mathbf{C}^+\boldsymbol{w}(k) - \mu\mathbf{C}^T\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}(k) - \mu\mathbf{C}^T\mathbf{C}\boldsymbol{\lambda}(k) \tag{23}$$

Therefore, we solve $\boldsymbol{\lambda}(k)$

$$\boldsymbol{\lambda}(k) = \frac{1}{\mu}(\mathbf{C}^T\mathbf{C})^{-1}(\mathbf{C}^T\boldsymbol{w}(k) - \mathfrak{F}) - (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}(k) \tag{24}$$

Plug this into (22), we get

$$\begin{aligned}\boldsymbol{w}(k+1) &= \boldsymbol{w}(k) - \mu\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}(k) - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\boldsymbol{w}(k) + \mu\mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}(k) + \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathfrak{F}\\ &= \boldsymbol{w}(k) - \mu[\mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T]\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}(k) + \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}[\mathfrak{F} - \mathbf{C}^T\boldsymbol{w}(k)]\end{aligned} \tag{25}$$

Notice that $\mathfrak{F} - \mathbf{C}^T\boldsymbol{w}(k)$ is not assumed to be zero, because at kth step, $\boldsymbol{w}(k)$ may not satisify the constraint, thus, we need to make it satisify. Define vector

$$\boldsymbol{f} = \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathfrak{F} \tag{26}$$

define $N \times N$ matrix

$$\mathbf{P} \triangleq \mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T \tag{27}$$

This is called projection matrix, we'll discull this in the next section. Then we rewrite (25) as

$$\boldsymbol{w}(k+1) = \mathbf{P}[\boldsymbol{w}(k) - \mu\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}(k)] + \boldsymbol{f} \tag{28}$$

However, until now, we still need covariance matrix to update weights, so a simple approximation $\boldsymbol{X}(k)\boldsymbol{X}(k)^+$ is used, so we use a different scheme to implement

$$\boldsymbol{w}(0) = \boldsymbol{f} \tag{29}$$

$$\boldsymbol{w}(k+1) = \mathbf{P}[\boldsymbol{w}(k) - \mu\boldsymbol{y}(k)\boldsymbol{X}(k)] + \boldsymbol{f} \tag{30}$$

By pre-multiplying $\boldsymbol{w}(k+1)$ by $\mathbf{C}^T$, we verify that

$$\begin{aligned}\mathbf{C}^T\boldsymbol{w}(k+1) &= \\ &= \mathbf{C}^T\{\boldsymbol{w}(k) - \mu[\mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T]\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}(k) + \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}[\mathfrak{F} - \mathbf{C}^T\boldsymbol{w}(k)]\} \\ &= \mathbf{C}^T\boldsymbol{w}(k) - \mu[\mathbf{C}^T - \mathbf{C}^T]\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w}(k) + [\mathfrak{F} - \mathbf{C}^T\boldsymbol{w}(k)] \\ &= \mathfrak{F} \qquad (31)\end{aligned}$$

Look back on the procedure, vector $\boldsymbol{f}$ and matrix $\mathbf{P}$ can be computed in advance, then in each step, all we need is some matrix multiplication and addition, in this way, computation is significantly reduced. The problem has a geometric interpretation. First of all, the linear constraint is composed of L linear equations about $\boldsymbol{w}$, then matrix defines a $N-L$ dimensional constraint plane, Γ,

$$\Gamma = \{\boldsymbol{w} : \mathbf{C}^T\boldsymbol{w} = \mathfrak{F}\} \qquad (32)$$

Fron linear algebra, we know vectors orthogonal to constraint plane is linear combination of columns of $\mathbf{C}$, namely, has the form $\mathbf{C}\boldsymbol{a}$, where $\boldsymbol{a}$ is a $L \times 1$ vector. And we observe that vector $\boldsymbol{f} = \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathfrak{F}$ has this form and thus orthogonal to constraint plane Γ, and also notice that point $\boldsymbol{f}$ is also on plane Γ, vector $\boldsymbol{f}$ starts from origin and ends at plane Γ, orthogonal to plane, the length of this vector is thus the distance from origin to this plane. We show this Fig. 5

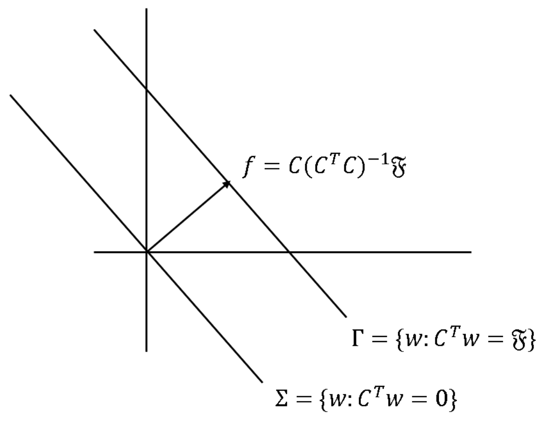

$f = C(C^TC)^{-1}\mathfrak{F}$
$\Gamma = \{w: C^Tw = \mathfrak{F}\}$
$\Sigma = \{w: C^Tw = 0\}$

Figure 5: The (N-L)-plane Γ and subspace Σ defined by the constraint

Pre-multiply any vector by matrix $\mathbf{P}$ will eliminate the component that orthogonal to constraint plane, therefore projecting the vector onto constraint subspace, where constraint subspace is defined by

$$\Sigma = \{\boldsymbol{w} : \mathbf{C}^T\boldsymbol{w} = 0\} \qquad (33)$$

This projection can be shown in Fig. 6

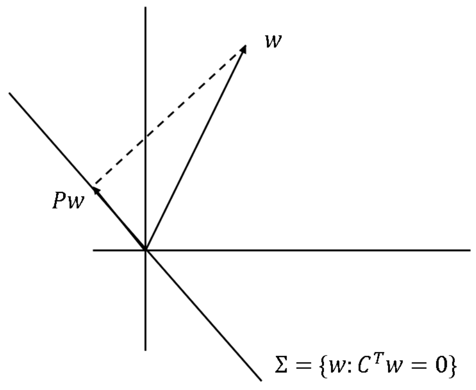


Figure 6: Matrix $\mathbf{P}$ projects vector onto the constraint subspace

The constraint is that every step, vector or point $\boldsymbol{w}(k)$ must be on plane Γ, so at step $k+1$, the new vector is not necessarily still on the constraint plane, we project the vector onto our constraint plane, and it is $\mathbf{P}[\boldsymbol{w}(k) - \mu \boldsymbol{y}(k)\boldsymbol{X}(k)]$, this vector is parallel to constraint subspaace, while vector $\boldsymbol{f}$ is still orthogonal to constraint subspace. Every step we update the component parallel to Σ, but leave orthogonal component untouched. This can be shown in Fig. 7

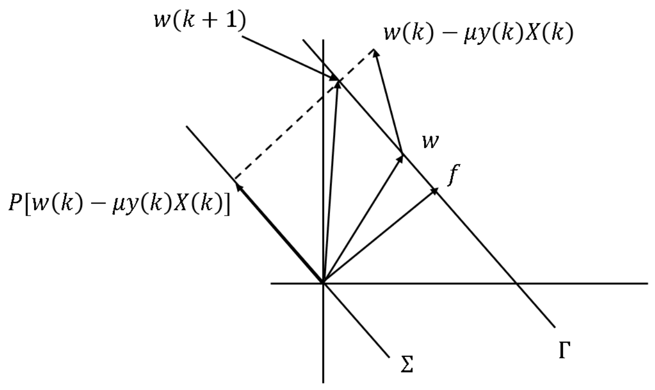


Figure 7: Update procedure of $\boldsymbol{w}(k+1) = \mathbf{P}[\boldsymbol{w}(k) - \mu \boldsymbol{y}(k)\boldsymbol{X}(k)] + \boldsymbol{f}$

In practice, all kinds of errors like round-off error, truncation or quantization error will cause drift from constraint plane, but projection makes sure that this error is eliminated at each step, so that it won't accumulate. This is the error correcting feature of this algorithm.

4.5 Improved Recursive Adaptive Beamforming

The algorithm by Frost has a problem that it is not robust to signal mismatch, Cox proposed improved algorithm. In [1], the array gain is defined as

$$G = \frac{\left|\boldsymbol{w}^{+}\boldsymbol{S}\right|^2}{\boldsymbol{w}^{+}\mathbf{R}_{\boldsymbol{N}}\boldsymbol{w}} \tag{34}$$

It is the gain due to beamforming, when we only have noise (no interference), it becomes white noise gain

$$G_w = \frac{\left|\boldsymbol{w}^+\boldsymbol{S}\right|^2}{\boldsymbol{w}^+\boldsymbol{w}} \tag{35}$$

The problem formulation is same with that of Frost, but with constraint on white noise gain, and it is a quadratic inequality constraint.

$$\underset{\boldsymbol{w}}{minimize} \quad \boldsymbol{w}^+\mathbf{R}_{\boldsymbol{X}}\boldsymbol{w} \tag{36}$$

$$\text{subject to } \mathbf{C}^+\boldsymbol{w} = \mathfrak{F} \tag{37}$$

$$G_w = \frac{\left|\boldsymbol{w}^+\boldsymbol{S}\right|^2}{\boldsymbol{w}^+\boldsymbol{w}} \geq \delta^2 \tag{38}$$

$$\boldsymbol{w}^+\boldsymbol{S} = 1 \tag{39}$$

We can see $\boldsymbol{w}^+\boldsymbol{S} = 1$ is just simplification of our problem, the quadratic constraint is not convex, therefore it can be transformed to convex constraint, but before we do transformation, we need to discuss projection matrix.

To project a point $\boldsymbol{a}$ in $\mathbb{R}^n$ onto the space spanned by matrix $\mathbf{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_k]$, where $C \in \mathbb{R}^{n\times k}$ and return the coordinate of that point, we start with projecting point a onto every basis of matrix space, assume matrix $\mathbf{C}$ is rank k, and $n > k$, coordinate of $\boldsymbol{a}$ represented by each column vector of $\mathbf{C}$ is $a_i = \frac{\boldsymbol{a}^T\boldsymbol{c}_i}{\|\boldsymbol{c}_i\|}$, and use matrix product form, $\boldsymbol{a}' = [\frac{\boldsymbol{c}_1}{\|\boldsymbol{c}_1\|}, \frac{\boldsymbol{c}_2}{\|\boldsymbol{c}_2\|}, \cdots, \frac{\boldsymbol{c}_k}{\|\boldsymbol{c}_k\|}]^T\boldsymbol{a}$, $\boldsymbol{a}' \in \mathbb{R}^{k\times 1}$. Now we need to use the coordinate of $\boldsymbol{a}$ in space $\mathbf{C}$ to represent the coordinate of the projection in original space. It is easy to know that by multiplying coordinate in space spanned by $\mathbf{C}$ by basis vectors in that space, which is represented by coordinate of original space, we can get the coordinate representation of this projection in original space, namely, $\boldsymbol{a}_{ori} = [\frac{\boldsymbol{c}_1}{\|\boldsymbol{c}_1\|}, \frac{\boldsymbol{c}_2}{\|\boldsymbol{c}_2\|}, \cdots, \frac{\boldsymbol{c}_k}{\|\boldsymbol{c}_k\|}]\boldsymbol{a}'$. Observe that $(\mathbf{C}^T\mathbf{C})^{-1} = diag(\frac{1}{\|\boldsymbol{c}_1\|^2}, \frac{1}{\|\boldsymbol{c}_2\|^2}, \cdots, \frac{1}{\|\boldsymbol{c}_k\|^2})$, so we have

$$\boldsymbol{a}_{ori} = \mathbf{C}diag(\frac{1}{\|\boldsymbol{c}_1\|^2}, \frac{1}{\|\boldsymbol{c}_2\|^2}, \cdots, \frac{1}{\|\boldsymbol{c}_k\|^2})\boldsymbol{a}' = \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\boldsymbol{a} = \mathbf{P}\boldsymbol{a} \tag{40}$$

Therefore, we can see that $\mathbf{P}$ projects a point into a subspace spanned by a matrix, and notice $\mathbf{PP} = \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T = \mathbf{P}$

We can decompose weight vector into two components, one is the projection in the constraint space, the other one is orthogonal to the constraint space, $\boldsymbol{w} = \boldsymbol{w}_c + \boldsymbol{v}$, where $\boldsymbol{v}$ is the one orthogonal to constraint space, we usually use another projection matrix to produce this matrix,

$$\tilde{\mathbf{P}} = \mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T \tag{41}$$

We have $\boldsymbol{w}_c = \mathbf{P}\boldsymbol{w}$, $\boldsymbol{v} = \tilde{\mathbf{P}}\boldsymbol{w}$, we can see $\boldsymbol{w}_c$ is the linear combination of columns of $\mathbf{C}$, and $\boldsymbol{v}$ is orthogonal to every column vector of $\mathbf{C}$. In the improved algorithm, we keep $\boldsymbol{w}^+\boldsymbol{S} = 1$, and therefore, quadratic constraint becomes

$$G_w = \frac{1}{\boldsymbol{w}_c^+\boldsymbol{w}_c + \boldsymbol{v}^+\boldsymbol{v}} \geq \delta^2 \tag{42}$$

because $\boldsymbol{w}_c^{+}\boldsymbol{v} = 0$, and according to optimal solution of $\boldsymbol{w}$, (21),

$$\begin{aligned} \boldsymbol{w}_c &= \mathbf{P}\boldsymbol{w} && (43)\\ &= \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{R}_{\boldsymbol{X}}^{-1}\mathbf{C}[\mathbf{C}^T\mathbf{R}_{\boldsymbol{X}}^{-1}\mathbf{C}]^{-1}\mathfrak{F} && (44)\\ &= \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathfrak{F} && (45) \end{aligned}$$

We can see that for any $\boldsymbol{w}$, $\boldsymbol{w}_c$ does not depend on $\boldsymbol{w}$, it is reasonable that for optimal beamformer, its component in constraint plane is same. Then $\boldsymbol{w}_c^{+}\boldsymbol{w}_c = \mathfrak{F}^{+}(\mathbf{C}^T\mathbf{C})^{-1}\mathfrak{F}$, then our constraint becomes

$$\boldsymbol{v}^{+}\boldsymbol{v} \leq \delta^{-2} - \mathfrak{F}^{+}(\mathbf{C}^T\mathbf{C})^{-1}\mathfrak{F} = b^2 \tag{46}$$

So the constraint is equivalent to a constraint on norm, and if we require constraint to be satisfied in each step, we can only update $\boldsymbol{v}$ because $\boldsymbol{w}_c$ always stays the same. Define the update vector

$$\boldsymbol{v}(k+1) = \tilde{\mathbf{P}}[\boldsymbol{v}(k) - \mu y(k)\boldsymbol{X}(k)] \tag{47}$$

We want constraint be satisfied in each step, so we normalize $\boldsymbol{v}$ in each step

$$\boldsymbol{w}(k+1) = \boldsymbol{w}_c + \begin{cases} \boldsymbol{v}(k+1) & for|\boldsymbol{v}|^2 \leq b^2 \\ \frac{b\boldsymbol{v}(k+1)}{|\boldsymbol{v}(k+1)|} & for|\boldsymbol{v}|^2 > b^2 \end{cases} \tag{48}$$

5 Simulations and Results

5.1 SMI Algorithm

Simulation of algorithms used MATLAB software, diagram of the program is shown in Fig. 8,

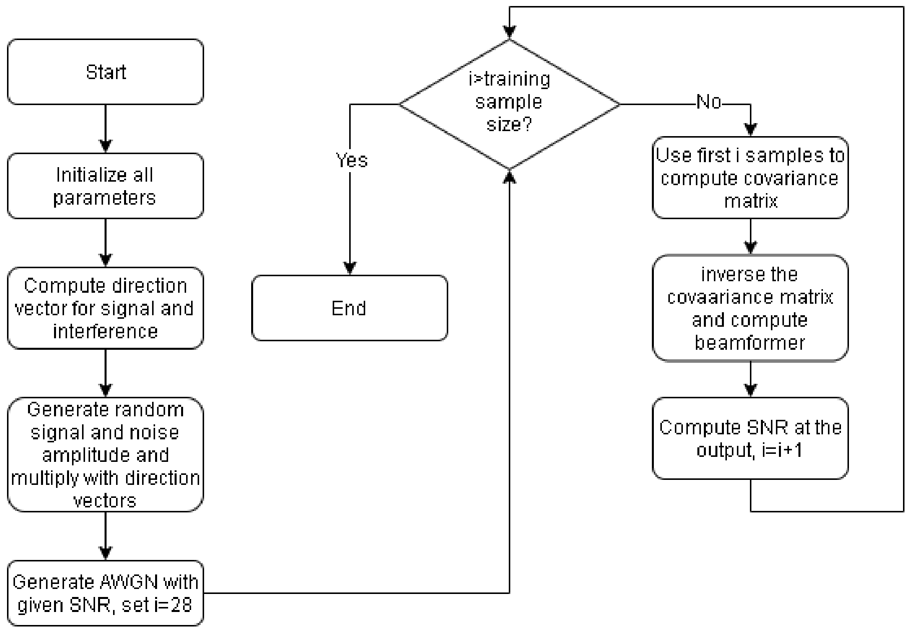

Start
Initialize all parameters
Compute direction vector for signal and interference
Generate random signal and noise amplitude and multiply with direction vectors
Generate AWGN with given SNR, set i=28
i>training sample size?
Yes
End
No
Use first i samples to compute covariance matrix
inverse the covaariance matrix and compute beamformer
Compute SNR at the output, i=i+1

Figure 8: Block diagram of SMI algorithm

To begin with, signal is represented as random variable times direction vector, which corresponds to real case,

$$\boldsymbol{S} = S\boldsymbol{h}_s = S[1, e^{j2\pi\frac{d}{\lambda}\sin\theta_s}, \cdots, e^{j2\pi(N-1)\frac{d}{\lambda}\sin\theta_s}] \tag{49}$$

θ_s is the angle where signal comes from. The representation of interference is same,

$$\boldsymbol{I}_1 = I_1\boldsymbol{h}_1 = I_1[1, e^{j2\pi\frac{d}{\lambda}\sin\theta_1}, \cdots, e^{j2\pi(N-1)\frac{d}{\lambda}\sin\theta_1}] \tag{50}$$

$$\boldsymbol{I}_2 = I_2\boldsymbol{h}_2 = I_2[1, e^{j2\pi\frac{d}{\lambda}\sin\theta_2}, \cdots, e^{j2\pi(N-1)\frac{d}{\lambda}\sin\theta_2}] \tag{51}$$

θ_1 and θ_2 are the direction of interference. The random variables have variance σ_s, σ_1 and σ_2, they are assumed to be Gaussian distributed, then they can be generated by random number generator. The next step is to compute the beamformer using samples, because if number of training samples is not enough, there will be problem to inverse the matrix, so we start with 28 samples, which is arbitrary.

The sample covariance matrix is then computed by first construct sample matrix $\mathbf{X}$, at ith step, we only use i samples to compute, and iteratively grow this number, in order to show the convergence $\hat{\mathbf{R}}_{\boldsymbol{N}} = \frac{1}{K}\mathbf{X}^T\mathbf{X}$. Therefore, $\boldsymbol{w} = \hat{\mathbf{R}}_{\boldsymbol{N}}^{-1}\boldsymbol{h}_s$. The SNR at the output can not be computed by (11) because we are using an estimation for covariance matrix. The results for SNR and antenna pattern are shown here

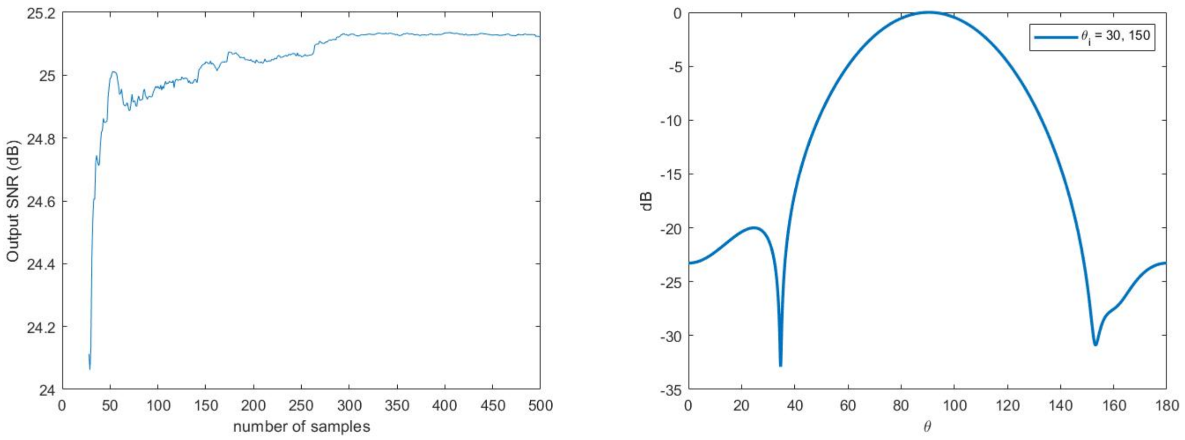


Figure 9: Output SNR (left) and pattern for the beamformer (right)

In the simulation, we set $N = 4$, $d = \frac{\lambda}{2}$, $\theta_s = 90°$,$\theta_1 = 30°$,$\theta_2 = 150°$, SNR is 20dB, 500 training samples, and $\sigma_s = 1$, $\sigma_1 = 0.5$, $\sigma_2 = 0.5$. As can be seen from the figure, as the number of training samples increases, output SNR increases first then stabilizes at a certain level, and it shows a very high SNR level, even higher than the case without interference. Besides, without beamforming, SINR of an isotropic antenna is 1/(0.25+0.25+0.01)=1.96, which is -1.79dB, this is due to the interference. And the second figure tells us that signals coming from direction $\theta_1 = 30°$ and $\theta_2 = 150°$ will suffer from significant attenuation, and they are just directions of interference. To investigate this property more, we use array with 5 elements, and make two sets of experiments, one set has interference coming from $30°, 150°$, another set has interference from $40°, 150°$, and here's what we have

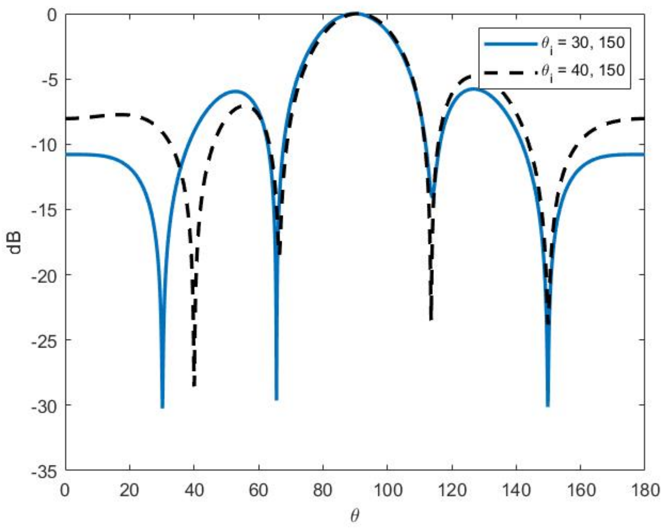


Figure 10: Antenna pattern with two different set of interferences

The generatio of antenna pattern used direction vector of all directions and do inner product with beamformer, after normalization, the response of beamformer to a direction will reflect the pattern.

5.2 Constrained LMS Algorithm

The program diagram is shown in Fig 11, we used similar direction vector as before, $N = 16 = ML$, $d = \frac{\lambda}{2}$, $\theta_s = 90°$,$\theta_1 = 30°$,$\theta_2 = 150°$, SNR is -10dB, 3000 training samples (same as in the paper), and $\sigma_s = \sqrt{0.1}$, $\sigma_1 = 1$, $\sigma_2 = 1$. As we can see, signal is weaker than interference and noise, but we let linear constraint $\mathfrak{F} = [1, 1, 1, 1]$ to guarantee look direction power. In order to simulate tapped filter, we need to construct linear constraint vectors exactly as in the paper, that is, $\mathbf{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4]$

$$\boldsymbol{c}_1^T = [1, e^{j2\pi\frac{d}{\lambda}\sin\theta_s}, e^{j4\pi\frac{d}{\lambda}\sin\theta_s}, e^{j6\pi\frac{d}{\lambda}\sin\theta_s}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \tag{52}$$
$$\boldsymbol{c}_2^T = [0, 0, 0, 0, 1, e^{j2\pi\frac{d}{\lambda}\sin\theta_s}, e^{j4\pi\frac{d}{\lambda}\sin\theta_s}, e^{j6\pi\frac{d}{\lambda}\sin\theta_s}, 0, 0, 0, 0, 0, 0, 0, 0] \tag{53}$$
$$\boldsymbol{c}_3^T = [0, 0, 0, 0, 0, 0, 0, 0, 1, e^{j2\pi\frac{d}{\lambda}\sin\theta_s}, e^{j4\pi\frac{d}{\lambda}\sin\theta_s}, e^{j6\pi\frac{d}{\lambda}\sin\theta_s}, 0, 0, 0, 0] \tag{54}$$
$$\boldsymbol{c}_4^T = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, e^{j2\pi\frac{d}{\lambda}\sin\theta_s}, e^{j4\pi\frac{d}{\lambda}\sin\theta_s}, e^{j6\pi\frac{d}{\lambda}\sin\theta_s}] \tag{55}$$
$$\tag{56}$$

After determining constraint matrix, we compute $\mathbf{C}$ and $\boldsymbol{f}$ by (26) and (27). Our direction vector can only have size same as number of elements, number of taps can not be considered because it will produce more nulls at the pattern. So instead of producing 16×1 sample vectors directly, we first generate four times the indicated number of samples and reshape samples to a 16 row sample matrix.Meanwhile, sample covariance matrix should still be computed because it's used in computation of output power (according to (14)).

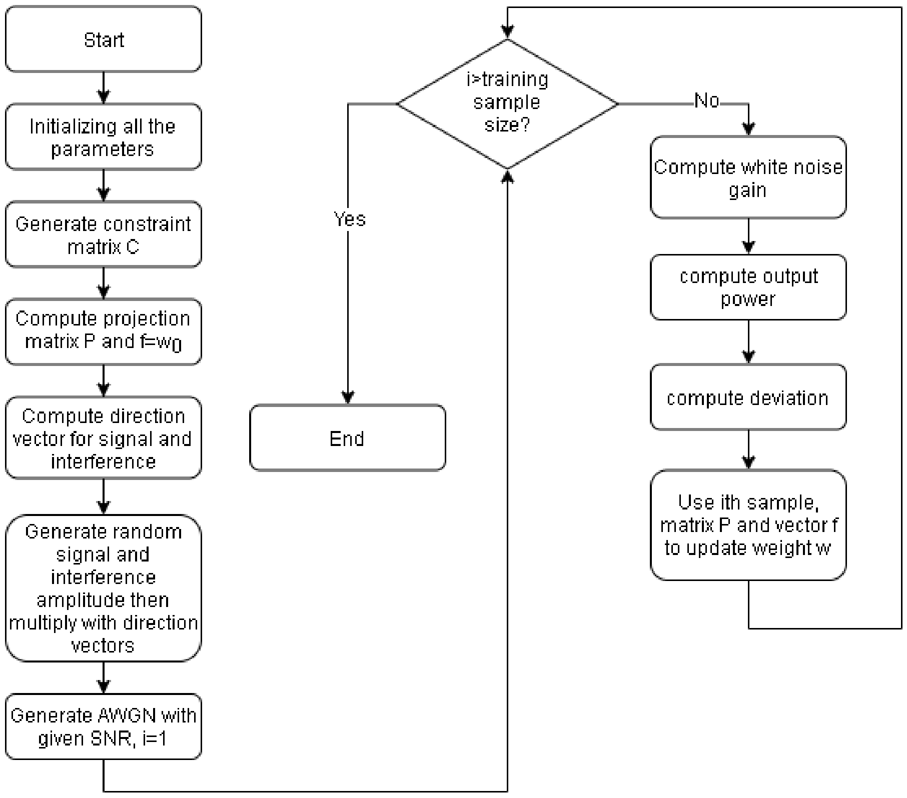

Start
Initializing all the parameters
Generate constraint matrix C
Compute projection matrix P and f=w0
Compute direction vector for signal and interference
Generate random signal and interference amplitude then multiply with direction vectors
Generate AWGN with given SNR, i=1
i>training sample size?
Yes
End
No
Compute white noise gain
compute output power
compute deviation
Use ith sample, matrix P and vector f to update weight w

Figure 11: Block diagram of LMS algorithm

During the iteration, at each update we first compute white noise gain (according to (35)), output power and deviation from constraint, deviation off constraint is given by

$$Deviation = \|\mathbf{C}^T \boldsymbol{w}(k) - \mathfrak{F}\|^2 \tag{57}$$

By inspecting output power, we have the following result

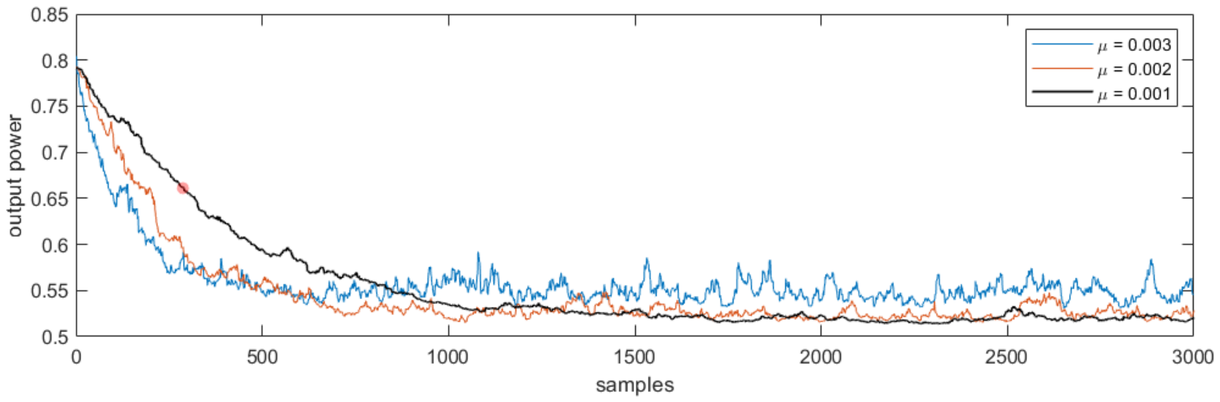


Figure 12: Output power with respect to different learning rate

As we can see from figure, with bigger learning rate ($\mu = 0.003$), the algorithm will converge faster, but at steady state it will also fluctuate and more unstable, and seems to be less optimal because output power higher than the other two case. If we choose low learning rate ($\mu = 0.001$), the algorithm converges slower but will better stabilize and achieve good performance, therefore, choice of learning rate is rather important.

The noise gain will be discussed in the next section together with improved LMS algorithm. We now examine the deviation off the constraint to see if round-off error will accumulate

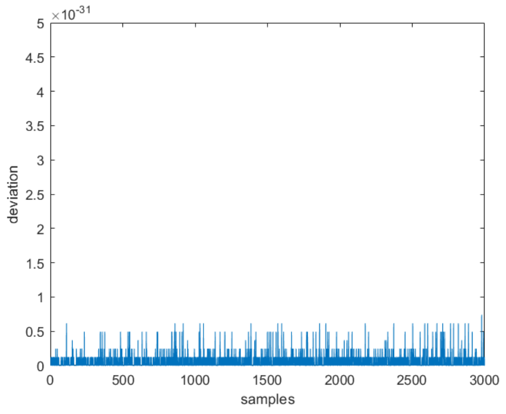
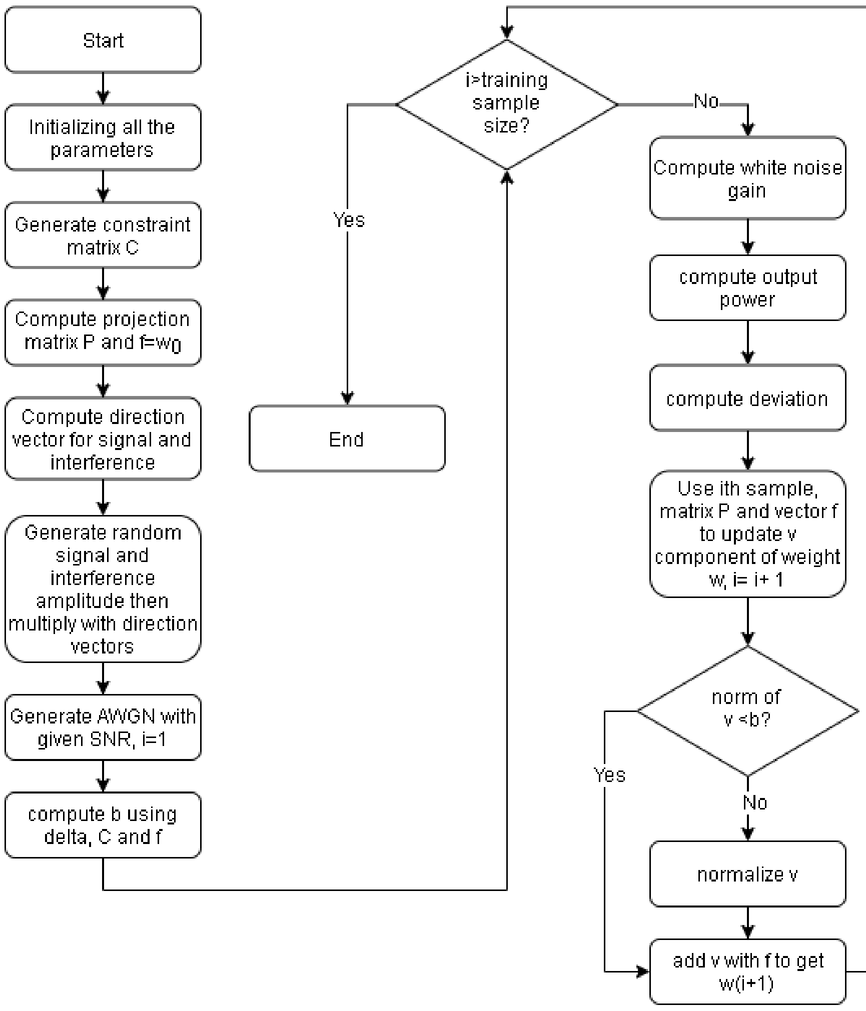


Figure 13: Deviation off the constraint

We should notice the scale is 10^{-31}, and the deviation is not growing, this means round-off errors are extremely small and error won't accumulate.



Start
Initializing all the parameters
Generate constraint matrix C
Compute projection matrix P and f=w0
Compute direction vector for signal and interference
Generate random signal and interference amplitude then multiply with direction vectors
Generate AWGN with given SNR, i=1
compute b using delta, C and f
i>training sample size?
Yes
No
End
Compute white noise gain
compute output power
compute deviation
Use ith sample, matrix P and vector f to update v component of weight w, i= i+ 1
norm of v <b?
Yes
No
normalize v
add v with f to get w(i+1)

Figure 14: Diagram of improved or robust LMS algorithm

5.3 Improved LMS Algorithm

The block diagram in Fig. 14 of this algorithm by Cox. is similiar to constrained LMS algorithm but with one more step at iteration process. Parameter setting is $\theta_s = 90°$,$\theta_1 = 30°$,$\theta_2 = 150°$, SNR is -10dB, 3000 training samples (same as in the paper), and $\sigma_s = \sqrt{0.1}$, $\sigma_1 = 1$, $\sigma_2 = 1$. Besides, $\delta = 0.98$ and $\mu = 0.002$. b in (46) is pre-computed, so it can be applied directly in the process. In each iteration, updating vector $\boldsymbol{v}(k+1)$ is computed and then its norm is compared to b, if the norm greater than b, we normalize it so that its norm equals b, if its norm smaller or equal than b, we do nothing and proceed. Because we only update $\boldsymbol{v}$, which is orthogonal to $\boldsymbol{f}$, therefore if we want to get $\boldsymbol{w}(k+1)$, we just need to add $\boldsymbol{v}(k+1)$ with $\boldsymbol{f}$.

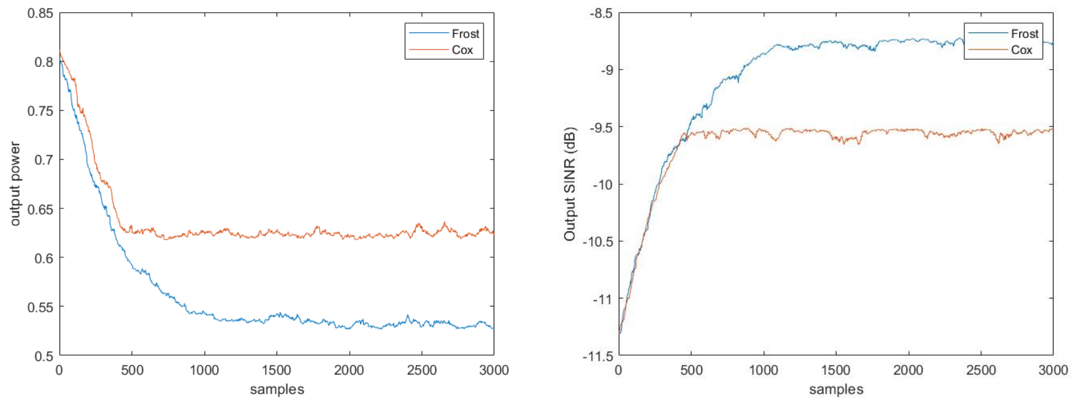


Figure 15: Output power (left) and SINR (right) of Frost algorithm and Cox algorithm

As we can see, both algorithms minimize output power as iteration proceeds, but for Cox algorithm, the output stops at a certain level and stabilize. This is when white noise gain comes into effect. However, the Frost algorithm has better SINR. White noise gain means our gain compared to only receiving white noise, it shows even if signal from non-broadside direction can still have some gain over white noise. It can be seen in the following figure

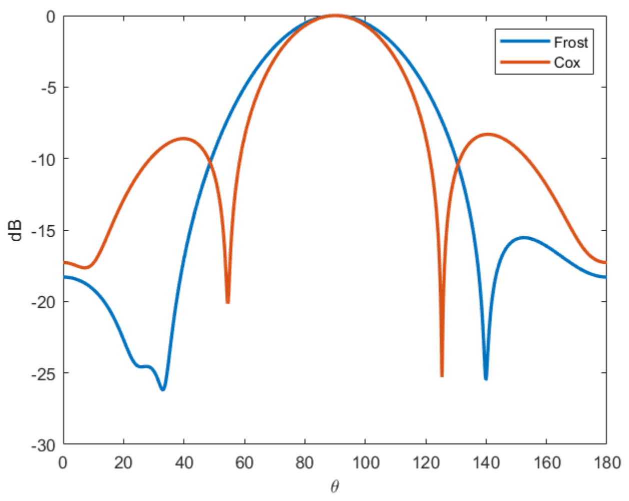
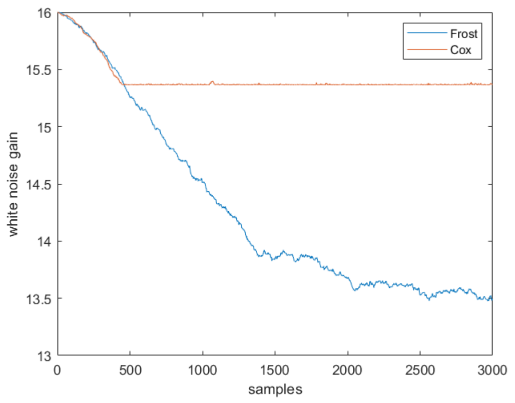


Figure 16: Antenna pattern of Frost algorithm and Cox algorithm

We can see that in Cox's beamformer, non-broadside suffers from less attenuation, this offers robustness against signal mismatch, because in real systems there will be imperfectness, might cause mismatch. We can compare the white noise gain of two algorithms,



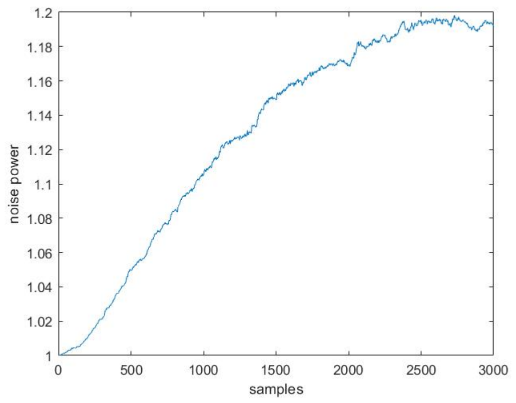


Figure 17: White noise gain of Frost algorithm and Cox algorithm (left), Output white noise power in Frost algorithm (right)

There is a tradeoff between output noise power and interference power, in Frost algorithm, interference power is minimized, but white noise power increases.

6 Conclusion and Remarks

In this report, we reviewed several papers that discuss optimization problems in beamforming, not only they have important influence in radar and sonar, they have impact in communication and very promising to become enabler for newest communication technologies. We started with algorithm by Reed, it converges faster than the other two, but it is computation demanding, and still requires enough samples to make inversion. The algorithm by Frost converges slowly, but has error correcting property, and it attenuates noise and interference dramatically. Cox's algorithm not only has error correcting feature, it also guanantees gain over white noise so that it's robust to signal mismatch, signal suppression won't be too terrible. The last two algorithms can allow the presence of signal, which is suitable for communication applications. In communication applications, we can change our model from tapped filter to simply antenna arrays, and all three algorithms discussed above requires quite a number of samples to guarantee performance, so directly apply to communication scenario is infeasible. In dely-demanding applications, we should avoid explicit matrix inversion, and for those applications require tracking user, we need faster converging algorithms and light computation.

Appendices

Appendix A

This is the code for SMI algorithm

```
clc
clear all

angle = 0:0.1:180;
rad = angle * 2 * pi / 360;
lamb = 4;
d = lamb / 2;
N = 4;

h0 = returnh(90, N, lamb, d);
h1 = returnh(30, N, lamb, d);
h2 = returnh(150, N, lamb, d);

hreal = returnh(70, N, lamb, d);

trainN = 500;
SNR=20;

alpha0 = 1 * randn(1,trainN);
alpha1 = 0.5 * randn(1,trainN);
alpha2 = 0.5 * randn(1,trainN);
sample1 = h1 * alpha1 + h2 * alpha2 ;
noisesample1 = awgn(sample1,SNR);
signalsample = h0 * alpha0;
SNRout = zeros(1, trainN);
SNRoutreal = zeros(1, trainN);
W = zeros(N, trainN);
strt = 28;
ngain = zeros(1, trainN);
for i = strt:trainN
noisesample1comp = noisesample1(:,1:i);
Rn1 = noisesample1comp * noisesample1comp' / i;
Rn1inv = inv(Rn1);
w1 = Rn1inv * h0;
W(:,i) = w1;
SNRout(i) = abs(W(:, i)' * h0)^2/ ...
abs(W(:, i)' * Rn1 * W(:, i));
SNRoutreal(i) = abs(W(:, i)' * hreal)^2/ ...
abs(W(:, i)' * Rn1 * W(:, i));
ngain(i) = abs(W(:, i)' * h0)^2 / norm(W(:,i))^2;
```

```
end

%% SNR with no mismatch
figure
plot(SNRout/max(SNRout),'DisplayName', 'no mismatch')

%% SNR with mismatch
hold on
plot(SNRoutreal/max(SNRout), 'DisplayName', 'signal mismatch')
ylim([0 1.5])
ylabel('Output SNR')
xlabel('number of samples')
legend

%% noise gain
% figure
% plot(ngain)

%% Antenna pattern
testh = returnh(angle, N, lamb, d);
res1 =W(:,500)' * testh;
res1 = abs(res1);
res1 = res1 / max(res1);

%% ULA pattern figure
figure
plot(angle, 10*log10(res1), 'LineWidth', 2, ...
'DisplayName', '\theta_i = 30, 150')
hold on
ylabel('dB')
xlabel('\theta')
legend

%% Generate direction vector
function [h] = returnh(angle, N, lamb, d)

rad = angle * 2 * pi / 360;
ind = [0: N-1].';
ph = 1i * 2 * pi * d * ind * cos(rad) / lamb;
h = exp(ph);

end
```

This is the code for Frost algorithm

```
clc
```

```
clear all

angle = 0:0.1:180;
rad = angle * 2 * pi / 360;
lamb = 4;
d = lamb / 2;
N = 4;
J = 4;

constraintangle = 90;
c1 = [ returnh(constraintangle, N, lamb, d)', zeros(1,12)]';
c2 = circshift(c1, 4);
c3 = circshift(c2, 4);
c4 = circshift(c3, 4);
C = [c1, c2, c3, c4];

f = [1, 1, 1, 1]';
F = C * inv(C' * C) * f;
P = eye(J * N) - C * inv(C' * C) * C';

trainN = 12000;
W0 = F;

realsigangle = 90;
interf1 = 30;
interf2 = 150;
h0 = returnh(realsigangle, N, lamb, d);
h1 = returnh(interf1, N, lamb, d);
h2 = returnh(interf2, N, lamb, d);
alpha0 = sqrt(0.1) * randn(1,trainN);
alpha1 = 1 * randn(1,trainN);
alpha2 = 1 * randn(1,trainN);

samp = h0 * alpha0 + h1 * alpha1 + h2 * alpha2;
noised = awgn(samp, 0, -10);

reshapesamp = [noised(:, 1: trainN/4); noised(:, trainN/4+1: trainN/2
noised(:, trainN/2+1: 3*trainN/4); noised(:, 3*trainN/4+1: trainN)];
Rx = reshapesamp * reshapesamp' * 4 / trainN;
W = [W0, zeros(16, trainN/4)];
power = zeros(1, trainN/4);
miu = 0.001;
ngain = zeros(1, trainN/4);
deviation = zeros(1, trainN/4);
for i = 1:trainN/4
```

```
ngain(i) = abs((c1 + c2 + c3 + c4)' * W(:, i))^2 ...
/ norm(W(:, i))^2;
power(i) = abs(W(:, i)' * Rx * W(:, i));
deviation(i) = norm(C' * W(:,i) - f)^2;
Wnew = P * (W(:, i) - miu * reshapesamp(:,i) * ...
reshapesamp(:,i)' * W(:, i)) + F;
W(:, i+1) = Wnew;
end

%% Deviation
% figure
% plot(deviation)
% ylim([0 0.00000000000000000000000000005])
% xlabel('samples')
% ylabel('deviation')

%% Output power
% figure
% plot(power, 'DisplayName', 'Frost')

%% White noise gain
% Run this code then run Cox algorithm code to compare in the same fi
figure
plot(ngain, 'DisplayName', 'Frost')
xlabel('samples')
ylabel('white noise gain')
hold on

%% Generate antenna pattern
testh = returnh(angle, N, lamb, d);
testh = [testh;testh;testh;testh];
w1 = W(:,end);
res1 = w1' * testh;
res1 = abs(res1);
res1 = res1 / max(res1);
%% Draw figure of antenna pattern
% figure
% plot(angle, 10*log10(res1), 'LineWidth', 2, ...
%      'DisplayName', 'end')
% hold on
% plot(angle, 10*log10(res2), 'LineWidth', 2, ...
%      'DisplayName', 'Frost')
% hold on
% legend
```

```
%% Generate direction vector
function [h] = returnh(angle, N, lamb, d)

rad = angle * 2 * pi / 360;
ind = [0: N-1].';
ph = 1i * 2 * pi * d * ind * cos(rad) / lamb;
h = exp(ph);

end
```

This is code for Cox algorithm

```
clc
clear all

angle = 0:0.1:180;
rad = angle * 2 * pi / 360;
lamb = 4;
d = lamb / 2;
N = 4;
J = 4;

constraintangle = 90;
c1 = [ returnh(constraintangle, N, lamb, d)', zeros(1,12)]';
c2 = circshift(c1, 4);
c3 = circshift(c2, 4);
c4 = circshift(c3, 4);
C = [c1, c2, c3, c4];

f = [1, 1, 1, 1]';
F = C * inv(C' * C) * f;
P = eye(J * N) - C * inv(C' * C) * C';

trainN = 12000;
W0 = F;

realsigangle = 90;
interf1 = 30;
interf2 = 150;
h0 = returnh(realsigangle, N, lamb, d);
h1 = returnh(interf1, N, lamb, d);
h2 = returnh(interf2, N, lamb, d);
alpha0 = sqrt(0.1) * randn(1,trainN);
alpha1 = 1 * randn(1,trainN);
alpha2 = 1 * randn(1,trainN);
```

```
samp = h0 * alpha0 + h1 * alpha1 + h2 * alpha2;
noised = awgn(samp, 0, -10);

reshapesamp = [noised(:, 1: trainN/4); noised(:, trainN/4+1: trainN/2
noised(:, trainN/2+1: 3*trainN/4); noised(:, 3*trainN/4+1: trainN)];
Rx = reshapesamp * reshapesamp' * 4 / trainN;
W = [W0, zeros(16, trainN/4)];
power = zeros(1, trainN/4);
ngain = zeros(1, trainN/4);
miu = 0.001;
delta = 0.98;
b2 = 1 / (delta)^2 - f' * inv(C' * C) * f;
b = sqrt(b2);
for i = 1:trainN/4
power(i) = abs(W(:, i)' * Rx * W(:, i));
ngain(i) = abs((c1 + c2 + c3 + c4)' * W(:, i))^2 ...
/ norm(W(:, i))^2;
vnew = P * (W(:, i) - F - miu * reshapesamp(:,i) * ...
reshapesamp(:,i)' * W(:, i));
absv = norm(vnew);
if absv^2 <= b2

elseif absv^2 > b2
vnew = b * vnew / absv;
end
W(:, i+1) = F + vnew;
end

%% White noise gain, run Frost algorithm code then run this
%figure
%plot(10*log10(ngain))
% xlabel('samples')
% ylabel('white noise gain')

%% Output power, run Frost algorithm code then run this
plot(power,'DisplayName', 'Cox')
xlabel('samples')
ylabel('output power')
hold on

%% Generate antenna pattern
testh = returnh(angle, N, lamb, d);
testh = [testh;testh;testh;testh];
w1 = W(:,end);
res1 = w1' * testh;
```

```
res1 = abs(res1);
res1 = res1 / max(res1);
%% Draw figure of antenna pattern
% figure
% plot(angle, 10*log10(res1), 'LineWidth', 2, ...
%       'DisplayName', 'Cox')
% legend
% ylabel('dB')
% xlabel('\theta')

%% Generate direction vector
function [h] = returnh(angle, N, lamb, d)

rad = angle * 2 * pi / 360;
ind = [0: N-1].';
ph = 1i * 2 * pi * d * ind * cos(rad) / lamb;
h = exp(ph);

end
```

References

[1] H. Cox, R. Zeskind, and M. Owen. Robust adaptive beamforming. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(10):1365–1376, 1987.

[2] O. L. Frost. An algorithm for linearly constrained adaptive array processing. *Proceedings of the IEEE*, 60(8):926–935, 1972.

[3] Z. Lian, L. Jiang, C. He, and D. He. User grouping and beamforming for hap massive mimo systems based on statistical-eigenmode. *IEEE Wireless Communications Letters*, 8(3):961–964, 2019.

[4] I. S. Reed, J. D. Mallett, and L. E. Brennan. Rapid convergence rate in adaptive arrays. *IEEE Transactions on Aerospace and Electronic Systems*, AES-10(6):853–863, 1974.

[5] S. A. Vorobyov, A. B. Gershman, and Zhi-Quan Luo. Robust adaptive beamforming using worst-case performance optimization: a solution to the signal mismatch problem. *IEEE Transactions on Signal Processing*, 51(2):313–324, 2003.