Chapter 3

Modified Algorithm

3.1 Quantized Weighting Method

From the previous chapter, we learn some TEQ algorithm. However, current design method such as MMSE, maximum shortening SNR (MSSNR), and maximum geometric SNR (MGSNR) do not maximize the bit rate directly. Since the bit rate is a function of noise, channel gain, and transmit power spectrum, a bit rate optimal TEQ design method must take into account all three. From the chapter two, we know only the maximum bit rate (MBR) method and Minimize-ISI (Min-ISI) method pay attention to maximize the bit rate.

By observing the Min-ISI cost function in equation (2.47), we can find that the weighting factor $\sum_{i=1}^{\bar{N}}(q_i \frac{S_{x,i}}{S_{n,i}} q_i^H)$ needs so many multiplication when $\bar{N}$ is large. From the viewpoint of real time implementation, in order to lower the computation complexity, we should reduce the multiplication. The way we proposed is to quantize the weighting factor. At first, we would like to set a SNR threshold. If the SNR $\frac{S_{x,i}}{S_{n,i}}$ is lower than the threshold, we will set it to "zero". The weighting being set to OFF means the subchannel is in bad situation. Since the noise power is too strong, we do

not even care about the ISI in this subchannel. Hence, it is reasonable to turn it off. On the contrary, if the situation that the SNR is higher than the threshold, we would like to set it to "one".

The threshold that we choose is to pick the subchannels which can transport more than three bits. By the Shannon channel capacity theorem, we can derive a value of signal-to-noise ratio that make these subcarriers take more than three bits.

Because the "Off & On" simplified method is rough, we would like to improve the accuracy. Based on the idea of quantization, we can extend the "Off & On" simplified method to multiple thresholds. However, from a point of view of the hardware implement, we still do not want to increase the load of the hardware. It implies that we do not look forward to raising the number of multiplication. As a result, our idea is to set four level thresholds, and they are "1", "2", "4", and "8". In this way, it only needs two comparisons. Besides, because these coefficients are power of 2, we can implement the multiplication by shifting. It really reduces the computation complexity.

3.2 Generalized Eigenvalue Problem

In the Min-ISI design method, the problem we deal with is as follow:

$$\arg\min_{w} \left(w^T X w\right) \quad \text{s.t.} \quad w^T Y w = 1$$

where

$$X = \mathrm{H}^T \mathrm{D}^{\mathrm{T}} \sum_{i \in S} (q_i \frac{S_{x,i}}{S_{n,i}} q_i^H) \mathrm{DH}$$

$$Y = \mathrm{H}^{\mathrm{T}} \mathrm{G}^{\mathrm{T}} \mathrm{GH}$$

We would like to figure out the problem by using Lagrange multiplier. We write the cost function as

$$L(w, l) = w^T Xw + l(w^T Yw - 1) \tag{3.1}$$

where l is a Lagrange multiplier. Before we deal with this cost function, we want to introduce a Lemmas and a Theorem.

Lemmas A: Let $x \in R^n$ be a vector.

Then

$$\frac{\partial}{\partial x_k} x_i x_j = \begin{cases} 2x_k & \text{for } i = j = k \\ x_j & \text{for } i = k \text{ and } j \neq k \\ x_i & \text{for } j = k \text{ and } i \neq k \\ 0 & \text{otherwise} \end{cases}$$

Theorem A: Let $A \in \mathrm{C}^n \times \mathrm{C}^n$ be an $n \times n$ array and $x \in \mathrm{R}^n$ be an n element vector. Then

$$\frac{\partial (x^T Ax)}{\partial x} = Ax + A^T x$$

Proof:

$$\frac{\partial (x^T Ax)}{\partial x} = \nabla_x (x^T Ax) = \nabla_x \left(\begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \right)$$

$$= \nabla_x \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i^* x_j = \begin{bmatrix} \frac{\partial}{\partial x_1} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i^* x_j \\ \frac{\partial}{\partial x_2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i^* x_j \\ \vdots \\ \frac{\partial}{\partial x_n} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i^* x_j \end{bmatrix} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \begin{bmatrix} \frac{\partial}{\partial x_1} (x_i^* x_j) \\ \frac{\partial}{\partial x_2} (x_i^* x_j) \\ \vdots \\ \frac{\partial}{\partial x_n} (x_i^* x_j) \end{bmatrix}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\begin{bmatrix} d_{i1}x_j + d_{j1}x_i \\ d_{i2}x_j + d_{j2}x_i \\ \vdots \\ d_{in}x_j + d_{jn}x_i \end{bmatrix} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_j\begin{bmatrix} d_{i1} \\ d_{i2} \\ \vdots \\ d_{in} \end{bmatrix} + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i\begin{bmatrix} d_{j1} \\ d_{j2} \\ \vdots \\ d_{jn} \end{bmatrix}$$

$$= \sum_{j=1}^{n} x_j \sum_{i=1}^{n} a_{ij}\begin{bmatrix} d_{i1} \\ d_{i2} \\ \vdots \\ d_{in} \end{bmatrix} + \sum_{i=1}^{n} x_i \sum_{j=1}^{n} a_{ij}\begin{bmatrix} d_{j1} \\ d_{j2} \\ \vdots \\ d_{jn} \end{bmatrix} = \sum_{j=1}^{n} x_j \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{nj} \end{bmatrix} + \sum_{i=1}^{n} x_i \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix}$$

$$= Ax + (x^T A)^T$$

$$= Ax + A^T x$$

If we take the gradient of $L(w, l)$ with respect to w and set it to the zero. That is

$$\frac{\partial L(w, l)}{\partial w} = 2Xw + 2lYw = 0 \tag{3.2}$$

The equation (3.2) is derived from the theorem A. From (3.2), we know that we can find an optimal w_{opt} to satisfy it. That is

$$2Xw_{opt} = -2lYw_{opt} \tag{3.3}$$

This is the generalized eigenvalue and eigenvector problem. However, the problem we usually deal with is the regular eigenvalue problem. Hence, we may prefer transferring (3.3) to another form. We will eliminate the coefficient "2", and set $\tilde{l} = -l$. We will obtain

$$w_{opt} = \tilde{l}\, X^{-1}Yw_{opt}$$

$$\frac{1}{\tilde{l}} w_{opt} = (X^{-1}Y)w_{opt}$$

$$\hat{l}\, w_{opt} = Mw_{opt} \tag{3.4}$$

By observing (3.4), we can find it is a regular eigenvalue problem.

In (3.3), the optimal solution is the eigenvector corresponding to the minimum eigenvalue. Moreover, the minimum generalized eigenvalue $\tilde{l}$ of the matrix pair *X* and *Y* is equivalent to the maximum eigenvalue $\hat{l}$ of the matrix *M*. Therefore, we only need to compute the maximum eigenvalue.

Following, we will adopt the power method to get the maximum eigenvalue. After several iterative computations, we can derive to the dominant (maximum) eigenvalue and its corresponding eigenvector. The power method steps are as follows:

Start with the vector

(1)

$$w_0 = [111\cdots 1]'.$$

Generate the sequence $\{w_k\}$ recursively, using

(2)

$$Q_k = Mw_k,$$

$$w_{k+1} = \frac{1}{c_{k+1}} Q_k$$

where c_{k+1} is the coordinate of Q_k of largest magnitude (in the case of a tie, choose the coordinate that comes first). The sequences $\{w_k\}$ and $\{c_k\}$ will converge to V and $\hat{l}$, respectively:

(3)

$$\lim_{k\to\infty} x_k = V \quad \text{and} \quad \lim_{k\to\infty} c_k = \hat{l}$$

Since we only care about *w*, the $\hat{l}$ does not have to be calculated.