GPU Implementation of Data-Aided Equalizers

Jeffrey T. Ravert

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Michael D. Rice, Chair
Brian D. Jeffs
Brian A. Mazzeo

Department of Electrical and Computer Engineering

Brigham Young University

April 2017

ABSTRACT

GPU Implementation of Data-Aided Equalizers

Jeffrey T. Ravert

Department of Electrical and Computer Engineering

Master of Science

Multipath is one of the dominant causes for link loss in aeronautical telemetry. Equalizers have been studied to combat multipath interference in aeronautical telemetry. Blind Constant Modulus Algorithm (CMA) equalizers are currently being used on SOQPSK-TG. The Preamble Assisted Equalization (PAQ) has been funded by the Air Force to study data-aided equalizers on SOQPSK-TG. PAQ compares side by side no equalization, data-aided zero forcing equalization, data-aided MMSE equalization, data-aided initialized CMA equalization, data-aided frequency domain equalization, and blind CMA equalization. An real time experimental test setup has been assembled including an RF receiver for data acquisition, FPGA for hardware interfacing and buffering, GPUs for signal processing, spectrum analyzer for viewing multipath events, and an 8 channel bit error rate tester to compare equalization performance. Lab tests were done with channel and noise emulators. Flight tests were conducted in March 2016 and June 2016 at Edwards Air Force Base to test the equalizers on live signals. The test setup achieved a 10Mbps throughput with a 6 second delay. Counter intuitive to the simulation results, the flight tests at Edwards AFB in March and June showed blind equalization is superior to data-aided equalization. Lab tests revealed some types of multipath caused timing loops in the RF receiver to produce garbage samples. Data-aided equalizers based on data-aided channel estimation leads to high bit error rates. A new experimental setup is been proposed, replacing the RF receiver with a RF data acquisition card. The data acquisition card will always provide good samples because the card has no timing loops, regardless of severe multipath.

Keywords: MISSING

ACKNOWLEDGMENTS

Students may use the acknowledgments page to express appreciation for the committee members, friends, or family who provided assistance in research, writing, or technical aspects of the dissertation, thesis, or selected project. Acknowledgments should be simple and in good taste.

Table of Contents

List of Tables

List of Figures

Chapter 1

System Overview

1.1 Overview

This chapter gives a high lever overview of the the algorithms implemented into the GPU. A block diagram is shown in Figure 1.1. The algorithms implemented in GPUs will briefly be explained. Chapter 2 explains the computation and application of the equalizers at a lower level. A simple block Diagram is shown in Figure 1.1.

This chapter will proceed as follows, section 1.1.1 will explain the algorithm used to find the preambles and packetize the received signal, section **??** will explain the frequency offset estimator and frequency offset compensation, section 1.1.2 will explain channel channel estimation, section 1.1.3 will explain noise variance, section 1.1.4 will explain the GPU implementation of the OQPSK detector. The explanation of the GPU implantation of the equalizers will be explained in much detail in Chapter 2.

1.1.1 Preamble Detection

The received samples in this project has the iNET packet structure shown in Figure 1.2. The iNET packet consists of a preamble and ASM periodically inserted into the data stream. The iNET preamble and ASM bits are inserted every 6144 data bits. The received signal is sampled at 2 samples/bit, making a iNET packet L_{pkt} long or 12672 samples. The iNET preamble comprises eight repetitions of the 16-bit sequence CD98_{hex} and the ASM field

$$034776C72728950B0_{\text{hex}} \tag{1.1}$$

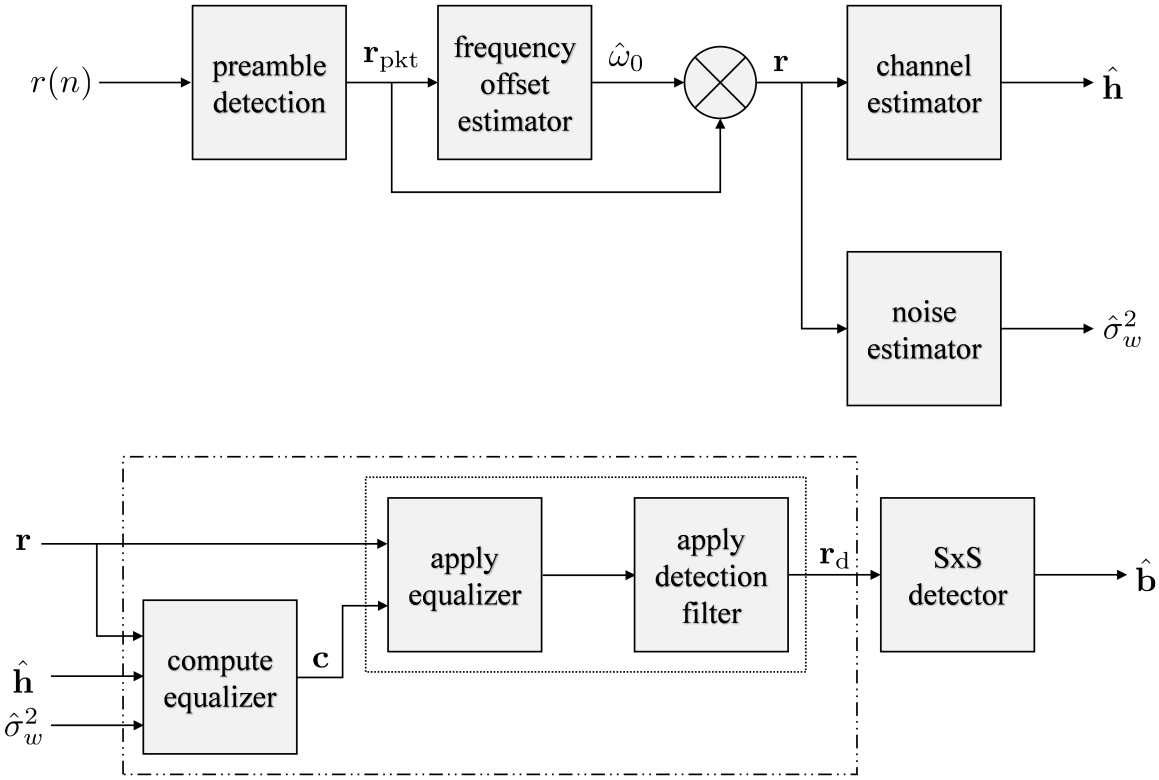

Figure 1.1: This a simple block diagram of what the GPU does.

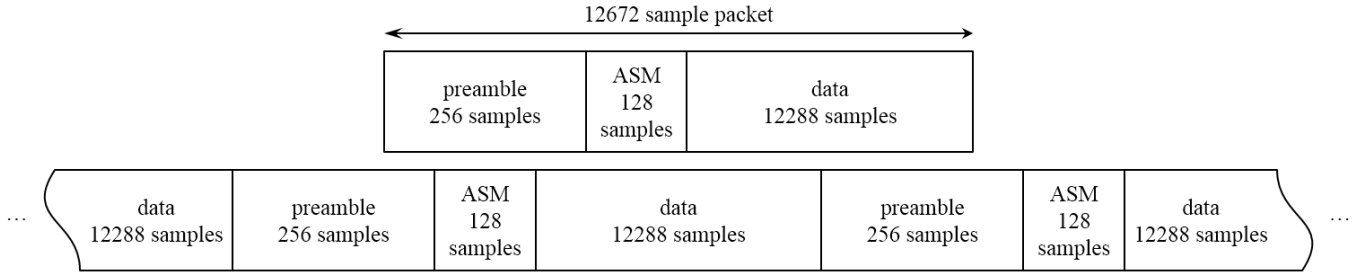

Figure 1.2: The iNET packet structure.

To compute data-aided preamble assisted equalizers, preambles in the received signal are found then used to estimate various parameters. The goal of the preamble detection step is to "packetize" the received samples into vectors with the packet structure shown in Figure 1.2. Each packet of received samples contains a L_{p} preamble samples, L_{ASM} ASM samples and L_{ASM} data samples.

Before the received samples can be packetized, the preambles are found using a preamble detector explained in [1]. The preamble detector output $L(u)$ is computed by

$$L(u) = \sum_{m=0}^{7} \left[I^2(n,m) + Q^2(n,m) \right] \tag{1.2}$$

where the inner summations are

$$\begin{aligned} I(n,m) \approx & \sum_{\ell \in \mathcal{L}_1} r_R(\ell+32m+n) - \sum_{\ell \in \mathcal{L}_2} r_R(\ell+32m+n) + \sum_{\ell \in \mathcal{L}_3} r_I(\ell+32m+n) - \sum_{\ell \in \mathcal{L}_4} r_I(\ell+32m+n) \\ & + 0.7071 \left[\sum_{\ell \in \mathcal{L}_5} r_R(\ell+32m+n) - \sum_{\ell \in \mathcal{L}_6} r_R(\ell+32m+n) \right. \\ & \left. + \sum_{\ell \in \mathcal{L}_7} r_I(\ell+32m+n) - \sum_{\ell \in \mathcal{L}_8} r_I(\ell+32m+n) \right], \end{aligned} \tag{1.3}$$

and

$$\begin{aligned} Q(n,m) \approx & \sum_{\ell \in \mathcal{L}_1} r_I(\ell+32m+n) - \sum_{\ell \in \mathcal{L}_2} r_I(\ell+32m+n) \\ & - \sum_{\ell \in \mathcal{L}_3} r_R(\ell+32m+n) + \sum_{\ell \in \mathcal{L}_4} r_R(\ell+32m+n) \\ & + 0.7071 \left[\sum_{\ell \in \mathcal{L}_5} r_I(\ell+32m+n) - \sum_{\ell \in \mathcal{L}_6} r_I(\ell+32m+n) \right. \\ & \left. - \sum_{\ell \in \mathcal{L}_7} r_R(\ell+32m+n) + \sum_{\ell \in \mathcal{L}_8} r_R(\ell+32m+n) \right] \end{aligned} \tag{1.4}$$

with

$$\begin{aligned}
\mathcal{L}_1 &= \{0, 8, 16, 24\} \\
\mathcal{L}_2 &= \{4, 20\} \\
\mathcal{L}_3 &= \{2, 10, 14, 22\} \\
\mathcal{L}_4 &= \{6, 18, 26, 30\} \\
\mathcal{L}_5 &= \{1, 7, 9, 15, 17, 23, 25, 31\} \\
\mathcal{L}_6 &= \{3, 5, 11, 12, 13, 19, 21, 27, 28, 29\} \\
\mathcal{L}_7 &= \{1, 3, 9, 11, 12, 13, 15, 21, 23\} \\
\mathcal{L}_8 &= \{5, 7, 17, 19, 25, 27, 28, 29, 31\}.
\end{aligned} \tag{1.5}$$

Figure 1.3 shows $2L_{\text{pkt}}$ samples of the preamble detector output $L(u)$. The start of a preamble is indicated by a local maximum of the preamble detector output. Using the index of the local maximums, the received samples are packetized. The vector $\mathbf{r}_{\text{pkt}}$ as shown in Figure 1.1 contains 12672 samples of data with the packet structure shown in Figure 1.2.

The preamble detection algoritm in Equations (1.2)-(1.5) and the local maximum search algorithms are easily implemented into GPUs. The GPU implementation of these algorithms wont be explained here.

Frequency Offset Compensation

The frequency offset estimator shown in Figure 1.1 is an algorithm taken from blah. With the notation adjusted slightly, the frequency offset estimate is

$$\hat{\omega}_0 = \frac{1}{L_q} \arg \left\{ \sum_{n=i+2L_q}^{i+7L_q-1} r(n) r^*(n - L_q) \right\} \tag{1.6}$$

where a frequency offset estimate is produced for every packet in $\mathbf{r}_{\text{pkt}}$. Equation (1.6) is easily implemented into GPUs.

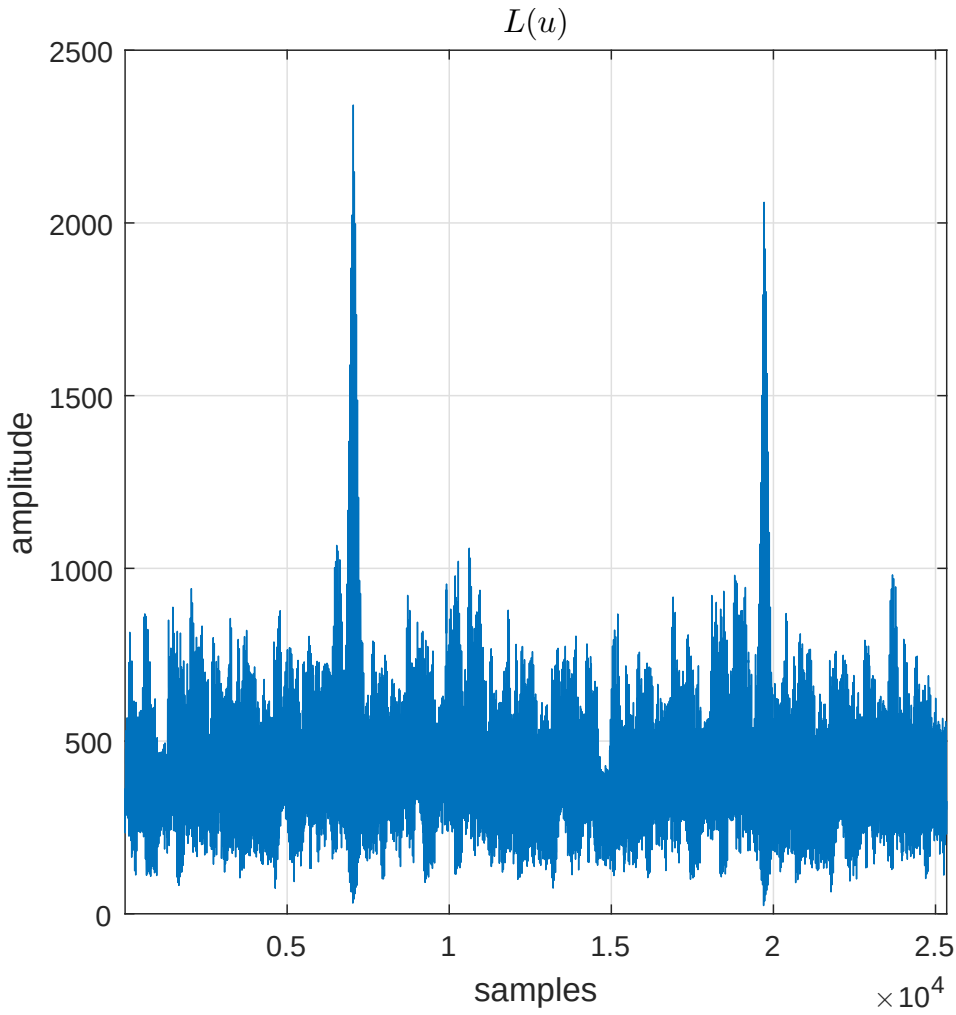

Figure 1.3: The output of the Preamble Detector $L(u)$.

1.1.2 Channel Estimation

1.1.3 Noise Variance Estimation

1.1.4 OQPSK Detector

Chapter 2

Equalizer Equations

2.1 Overview

There are 3 different kinds of equalizers I run 1. the solving ones!!! They are equations like Ax=b where I have A and b but I need x 2. the initialized then iterative ones. CMA is initialized with MMSE then runs as many times a possible 3. the multiply ones! the FDEs are a simple multiply in the frequency domain

2.2 Equations

2.2.1 The Solving Equalizers

The Zero-Forcing Equalizer

The ZF equalizer was studied in the PAQ Phase 1 Final Report in equation 324

$$\mathbf{c}_{\text{ZF}} = (\mathbf{H}^{\dagger}\mathbf{H})^{-1}\mathbf{H}^{\dagger}\mathbf{u}_{n_0} \tag{2.1}$$

where $\mathbf{c}_{\text{ZF}}$ is a $L_{eq} \times 1$ vector of equalizer coefficients computed to invert the channel estimate $\mathbf{h}$ and $\mathbf{u}_{n_0}$ is the desired channel impulse response centered on $n0 = N_1 + L_1 + 1$

$$\mathbf{u}_{n_0} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} n_0 - 1 \text{ zeros} \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} N_1 + N_2 + L_1 + L_2 - n_0 + 1 \text{ zeros} \end{matrix} . \tag{2.2}$$

The $L_{eq} + N_1 + N_2 \times L_{eq}$ convolution matrix $\mathbf{H}$ is built using the channel estimate $\mathbf{h}$

$$\mathbf{H} = \begin{bmatrix} h(-N_1) & & & \\ h(-N_1+1) & h(-N_1) & & \\ \vdots & \vdots & \ddots & \\ h(N_2) & h(N_2-1) & & h(-N_1) \\ & h(N_2) & & h(-N_1+1) \\ & & & \vdots \\ & & & h(N_2) \end{bmatrix}. \tag{2.3}$$

The computation of the coefficients in Equation (2.1) can be simplified in a couple of ways: First the matrix multiplication of $\mathbf{H}^\dagger$ and $\mathbf{H}$ is the autocorrelation matrix of the channel

$$\mathbf{R}_h = \mathbf{H}^\dagger\mathbf{H} = \begin{bmatrix} r_h(0) & r_h^*(1) & \cdots & r_h^*(L_{eq}-1) \\ r_h(1) & r_h(0) & \cdots & r_h^*(L_{eq}-2) \\ \vdots & \vdots & \ddots & \\ r_h(L_{eq}-1) & r_h(L_{eq}-2) & \cdots & r_h(0) \end{bmatrix} \tag{2.4}$$

where

$$r_h(k) = \sum_{n=-N_1}^{N_2} h(n)h^*(n-k). \tag{2.5}$$

Second the matrix vector multiplication of $\mathbf{H}^\dagger$ and $\mathbf{u}_{n_0}$ is simply the n_0th row of $\mathbf{H}^\dagger$ or the conjugated n_0th column of $\mathbf{H}$. A new vector $\mathbf{h}_{n_0}$ is defined by

$$\mathbf{h}_{n_0} = \mathbf{H}^\dagger\mathbf{u}_{n_0} = \begin{bmatrix} h(L_1) \\ \vdots \\ h(0) \\ \vdots \\ h(-L_2) \end{bmatrix}. \tag{2.6}$$

To simplify, Equations (2.4) and (2.6) are substituted into Equation (2.1) resulting in

$$\mathbf{c}_{\text{ZF}} = \mathbf{R}_h^{-1}\mathbf{h}_{n_0}. \tag{2.7}$$

Computing the inverse of $\mathbf{R}_h$ is computationally heavy because an inverse is an N^3 operation. To avoid an inverse, $\mathbf{R}_h$ is moved to the left side and $\mathbf{c}_{\text{ZF}}$ is found by solving a system of linear equations. Note that $r_h(k)$ only has support on $-L_{ch} \leq k \leq L_{ch}$ making $\mathbf{R}_h$ sparse or $\%63$ zeros. The sparseness of $\mathbf{R}_h$ is leveraged to reduce computation drastically. The Zero-Forcing Equalizer coefficients are computed by solving

$$\mathbf{R}_h\mathbf{c}_{\text{ZF}} = \mathbf{h}_{n_0}. \tag{2.8}$$

MMSE Equalizer

The MMSE equalizer has the same form as the Zero-Forcing equalizer. The MMSE equalizer was also studied in the PAQ Phase 1 Final Report in equation 330.

$$\mathbf{c}_{MMSE} = \left[\mathbf{G}\mathbf{G}^\dagger + \frac{\sigma_w^2}{\sigma_s^2}\mathbf{I}_{L_1+L_2+1}\right]\mathbf{g}^\dagger \tag{2.9}$$

where

$$\mathbf{G} = \begin{bmatrix} h(N_2) & \cdots & h(-N_1) & & & \\ & h(N_2) & \cdots & h(-N_1) & & \\ & & \ddots & & \ddots & \\ & & & h(N_2) & \cdots & h(-N_1) \end{bmatrix} \tag{2.10}$$

and

$$\mathbf{g} = \begin{bmatrix} h(L_1) \cdots h(-L_2) \end{bmatrix}. \tag{2.11}$$

The vector $\mathbf{g}^\dagger$ is also the same vector as $\mathbf{h}_{n_0}$ in Equation (2.2). The matrix multiplication $\mathbf{G}\mathbf{G}^\dagger$ is also the same autocorrelation matrix $\mathbf{R}_h$ as Equation (2.4). The fraction $\frac{1}{2\hat{\sigma_w^2}}$ is substituted in for the fraction $\frac{\sigma_w^2}{\sigma_s^2}$ using Equation 333 Rice's report. MMSE only differs from Zero-Forcing by adding the signal-to-noise ratio estimate down the diagonal of the autocorrelation matrix $\mathbf{R}_h$. Substituting

in all these similarities in to Equation (2.9) results in

$$\Big[\mathbf{R}_h + \frac{1}{2\hat{\sigma}_w^2}\mathbf{I}_{L_1+L_2+1}\Big]\mathbf{c}_{\text{MMSE}} = \mathbf{h}_{n_0}. \tag{2.12}$$

To further simplify the notation, $\mathbf{R}_{hw}$ is substituted in for $\mathbf{R}_h + \frac{1}{2\hat{\sigma}_w^2}\mathbf{I}_{L_1+L_2+1}$ where

$$\mathbf{R}_{hw} = \mathbf{R}_h + \frac{1}{2\hat{\sigma}_w^2}\mathbf{I}_{L_1+L_2+1} = \begin{bmatrix} r_h(0) + \frac{1}{2\hat{\sigma}_w^2} & r_h^*(1) & \cdots & r_h^*(L_{eq}-1) \\ r_h(1) & r_h(0) + \frac{1}{2\hat{\sigma}_w^2} & \cdots & r_h^*(L_{eq}-2) \\ \vdots & \vdots & \ddots & \\ r_h(L_{eq}-1) & r_h(L_{eq}-2) & \cdots & r_h(0) + \frac{1}{2\hat{\sigma}_w^2} \end{bmatrix}. \tag{2.13}$$

The MMSE equalizer coefficients are solved for in a similar fashion to the Zero-Forcing equalizer coefficients in Equation (2.8).

$$\mathbf{R}_{hw}\mathbf{c}_{\text{MMSE}} = \mathbf{h}_{n_0}. \tag{2.14}$$

2.2.2 The Iterative Equalizer

The Constant Modulus Algorithm

CMA uses a steepest decent algorithm.

$$\mathbf{c}_{b+1} = \mathbf{c}_b - \mu \nabla \mathbf{J} \tag{2.15}$$

The vector $\mathbf{J}$ is the cost function and ∇J is the cost function gradient defined in the PAQ report 352 by

$$\nabla J = \frac{2}{L_{pkt}} \sum_{n=0}^{L_{pkt}-1} \Big[y(n)y^*(n) - R_2 \Big] \, y(n)\mathbf{r}^*(n). \tag{2.16}$$

where

$$\mathbf{r}(n) = \begin{bmatrix} r(n+L_1) \\ \vdots \\ r(n) \\ \vdots \\ r(n-L_2) \end{bmatrix}. \tag{2.17}$$

This means ∇J is of the form

$$\nabla J = \begin{bmatrix} \nabla J(-L_1) \\ \vdots \\ \nabla J(0) \\ \vdots \\ \nabla J(L_2) \end{bmatrix}. \tag{2.18}$$

To leverage the computational efficiency of the Fast Fourier Transform (FFT), Equation (2.16) is re-expressed as a convolution.

To begin messaging ∇J

$$z(n) = 2 \Big[y(n)y^*(n) - R_2 \Big] \, y(n) \tag{2.19}$$

is defined to make the expression of ∇J to be

$$\nabla J = \frac{1}{L_{pkt}} \sum_{n=0}^{L_{pkt}-1} z(n)\mathbf{r}^*(n). \tag{2.20}$$

then writing the summation out in vector form

$$\nabla J = \frac{z(0)}{L_{pkt}} \begin{bmatrix} r^*(L_1) \\ \vdots \\ r^*(0) \\ \vdots \\ r^*(L_2) \end{bmatrix} + \frac{z(1)}{L_{pkt}} \begin{bmatrix} r^*(1+L_1) \\ \vdots \\ r^*(1) \\ \vdots \\ r^*(1-L_2) \end{bmatrix} + \cdots \frac{z(L_{pkt}-1)}{L_{pkt}} \begin{bmatrix} r^*(L_{pkt}-1+L_1) \\ \vdots \\ r^*(L_{pkt}-1) \\ \vdots \\ r^*(L_{pkt}-1-L_2) \end{bmatrix}. \tag{2.21}$$

The kth value of ∇J is

$$\nabla J(k) = \frac{1}{L_{pkt}} \sum_{m=0}^{L_{pkt}-1} z(m)r^*(m-k), \quad -L_1 \leq k \leq L_2. \tag{2.22}$$

The summation almost looks like a convolution. To put the summation in convolution form, define

$$\rho(n) = r^*(n). \tag{2.23}$$

Now

$$\nabla J(k) = \frac{1}{L_{pkt}} \sum_{m=0}^{L_{pkt}-1} z(m)\rho(k-m). \tag{2.24}$$

Because $z(n)$ has support on $0 \leq n \leq L_{pkt}-1$ and $\rho(n)$ has support on $-L_{pkt}+1 \leq n \leq 0$, the result of the convolution sum $b(n)$ has support on $-L_{pkt}+1 \leq n \leq L_{pkt}-1$. Putting all the pieces together, we have

$$\begin{aligned} b(n) &= \sum_{m=0}^{L_{pkt}-1} z(m)\rho(n-m) \\ &= \sum_{m=0}^{L_{pkt}-1} z(m)r^*(m-n) \end{aligned} \tag{2.25}$$

Comparing Equation (2.24) and (2.25) shows that

$$\nabla J(k) = \frac{1}{L_{pkt}} b(k), \quad -L_1 \leq k \leq L_2. \tag{2.26}$$

The values of interest are shown in Figure Foo!!!!(c)

This suggest the following algorithm for computing the gradient vector ∇J Matlab Code!!!

2.2.3 The Multiply Equalizers

The Frequency Domain Equalizer One

The Frequency Domain Equalizer One (FDE1) is the MMSE or wiener filter applied in the frequency domain. Ian E. Williams and M. Saquib derived FDE1 for this project in a paper called Linear Frequency Domain Equalization of SOQPSK-TG for Wideband Aeronautical Telemetry. The FDE1 equalizer is defined in Equation (11) as

$$C_{\text{FDE1}}(\omega) = \frac{\hat{H}^*(\omega)}{|\hat{H}(\omega)|^2 + \frac{1}{\hat{\sigma}^2}} \tag{2.27}$$

The term $C_{\text{FDE1}}(\omega)$ is the Frequency Domain Equalizer One frequency response at ω. The term $\hat{H}(\omega)$ is the channel estimate frequency response at ω. The term $\hat{\sigma}^2$ is the noise variance estimate, this term is completely independent of frequency because the noise is assumed to be white or spectrally flat.

FDE1 needs no massaging because Equation (2.27) is easily implemented in the GPU and it is computationally efficient.

The Frequency Domain Equalizer One

The Frequency Domain Equalizer Twe (FDE2) is the MMSE or wiener filter applied in the frequency domain. Ian E. Williams and M. Saquib derived FDE1 for this project in a paper called Linear Frequency Domain Equalization of SOQPSK-TG for Wideband Aeronautical Telemetry. The FDE2 equalizer is defined in Equation (12) as

$$C_{\text{FDE2}}(\omega) = \frac{\hat{H}^*(\omega)}{|\hat{H}(\omega)|^2 + \frac{\Psi(\omega)}{\hat{\sigma}^2}} \tag{2.28}$$

FDE2 almost identical to FDE1. The only difference is term $\Psi(\omega)$ in the denominator. The term $\Psi(\omega)$ is the average spectrum of SPQOSK-TG shown in Figure 2.1. FDE2 needs no massaging because Equation (2.28) is easily implemented in the GPU and is computationally efficient.

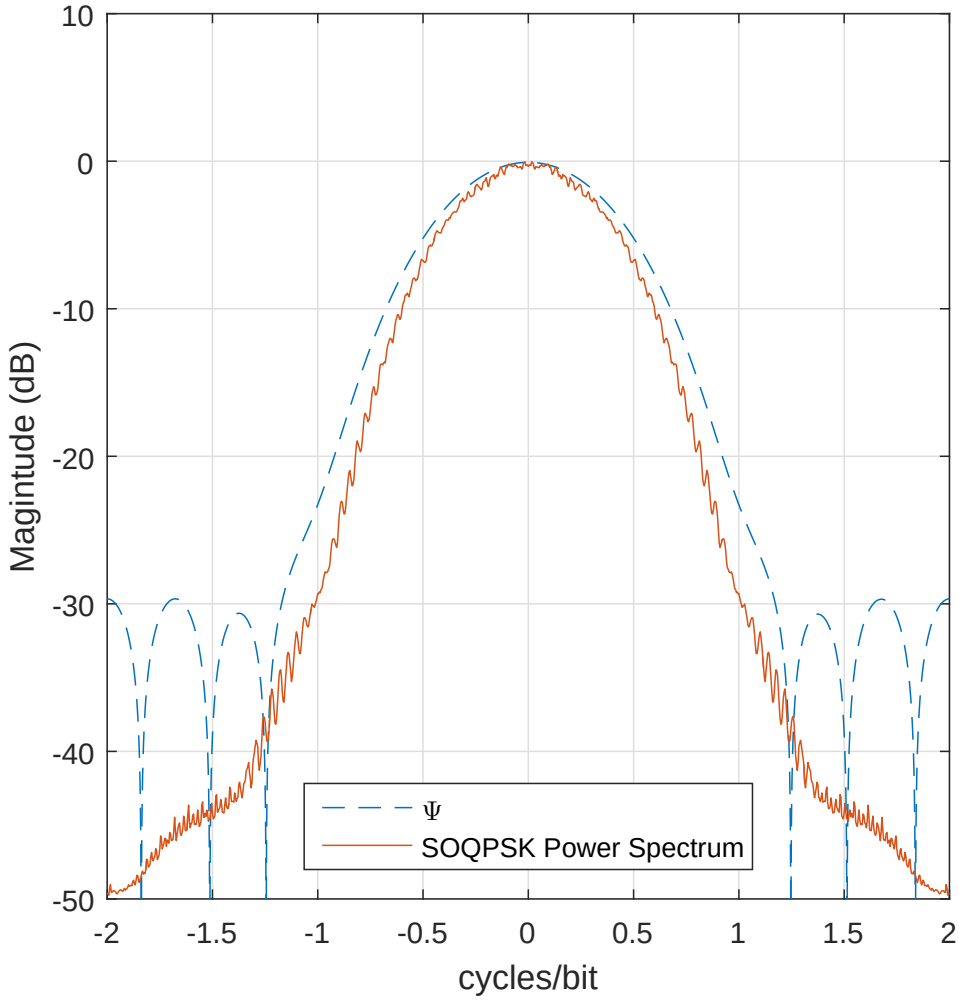

Figure 2.1: A block diagram illustrating organization of the algorithms in the GPU.

Bibliography

[1] M. Rice and A. Mcmurdie, "On frame synchronization in aeronautical telemetry," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 5, pp. 2263–2280, October 2016. 3