2017-06-01

A Survey of Sparse Channel Estimation in Aeronautical Telemetry

Christopher James Hogstrom
Brigham Young University

A Survey of Sparse Channel Estimation in Aeronautical Telemetry

Christopher James Hogstrom

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

Master of Science

Michael Rice, Chair
Neal K. Bangerter
David G. Long

Department of Electrical and Computer Engineering

Brigham Young University

ABSTRACT

A Survey of Sparse Channel Estimation in Aeronautical Telemetry

Christopher James Hogstrom
Department of Electrical and Computer Engineering, BYU
Master of Science

Aeronautical telemetry suffers from multipath interference, which can be resolved through the use of equalizers at the receiver. The coefficients of data-aided equalizers are computed from a channel estimate. Most channels seen in aeronautical telemetry are sparse, meaning that most of the coefficients of the channel are zero or nearly zero. The maximum likelihood (ML) estimate does not always produce a sparse channel estimate. This thesis surveys a number of sparse estimation algorithms that produce a sparse channel estimate and compares the post-equalizer bit error rates (BER) using these sparse estimates with the post-equalizer BER using the ML estimate. I show that the generalized Orthogonal Matching Pursuit (GOMP) performs the best followed by the Sparse Estimation based on Validation Re-estimated Least Squares (SPARSEVA-RE) and the Least Absolute Shrinkage and Selection Operator (LASSO).

Keywords: sparse channel estimation, multipath propagation, aeronautical telemetry, SOQPSK

ACKNOWLEDGMENTS

Foremost, I would like to express my thanks to my wife Kensie. Her support has helped me reach further than I previously thought possible and always motivated me when I felt like giving up.

I thank Dr. Michael Rice for his patience and guidance, I would not be where I am today without him. I owe him a few red pens.

Not to be forgotten, I thank my friends who kept me sane by playing racquetball weekly. Thank you Christopher Nash, Mitchell Burnett, and Jeff Ravert for explaining the difficult concepts we have learned through out our time as graduate students.

Finally, I would like to thank my parents for always encouraging me to further my education. Thank you for your love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Multipath in Aeronautical Telemetery

Aeronautical telemetry is the process of sending precise measurements about an airborne test article's condition to a ground station. This data is used for diagnostic purposes when something goes wrong, as well as to verify that the test article is functioning correctly during flight. Transmission of data occurs wirelessly, employing transmitters to send the data and receivers to recover the data. Fixed-wing and rotary-wing aircraft use a half-dipole antenna mounted on the underside of the fuselage. Missiles use "wrap-around" antennas – conformal antennas on the missile body.

Both types of antennas radiate the signal in all directions, allowing for multiple propagation paths from the transmitter to the receiver. The path directly from the transmitter to the receiver is called the line-of-sight path. All other paths are delayed, scaled versions of the line-of-sight component caused by reflectors in the propagation path. Figure 1.1 is a simple illustration of multipath interference. In the presence of multipath interference, the data is impossible to recover unless an equalizer is used at the receiver.

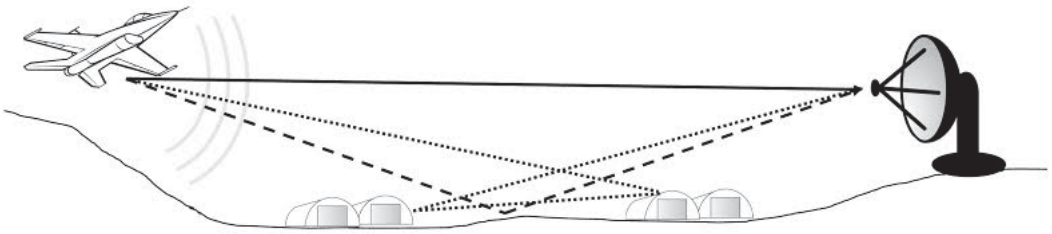


Figure 1.1: Mulitpath propagation in aeronautical telemetry. The solid line is the line-of-sight signal while the dashed lines are different propagation paths.

1.2 Equalizers

There are two types of equalizers that can be used to mitigate the effects of multipath influence, *blind* equalizers and *data-aided* equalizers. Blind equalizers are usually adaptive filters, and the filter adaptation criterion is usually based on some statistical property of the transmitted signal, such as constant modulus. Blind equalizers are agnostic to the data being received and do not form an estimate of the channel.

Data-aided equalizers are filters whose coefficients are computed using a channel estimate. The filter coefficients are computed based on minimizing a certain criteria. Commonly used criteria include, probability of error at the equalizer output, the intersymbol interference (ISI) at the equalizer output, or the mean squared error. All three of these criteria require knowledge of the channel seen by the equalizer. Knowledge of the channel is obtained using a "genie" (the true channel is perfectly known to the equalizer) or an estimate derived from the received data. To estimate the channel, a sequence of data, known to both the transmitter and receiver, is inserted into the telemetry downlink. This known data sequence is called a *pilot* sequence. Because equalizer performance (measured using the probability of error at the equalizer output) is affected by the quality of the channel estimate, high-quality channel estimates are important.

1.3 Problem Statement

Let $\mathbf{y}$ be the $q \times 1$ column vector formed by stacking the received waveform samples corresponding to the pilot sequence. This vector may be expressed as

$$\mathbf{y} = \mathbf{A}\mathbf{h} + \mathbf{z}, \tag{1.1}$$

where $\mathbf{A}$ is the $q \times Q$ ($q > Q$) convolution matrix formed from the waveform samples corresponding to the pilot sequence, $\mathbf{h}$ is the $Q \times 1$ vector formed from the (FIR) impulse response of the channel seen at the equalizer input, and $\mathbf{z}$ is a $q \times 1$ noise vector. The noise vector is modeled as

a complex-valued circularly symmetric Gaussian random vector with zero mean and covariance matrix $\sigma^2\mathbf{I}$.

Looking at Equation (1.1) from an estimation theory point of view, $\mathbf{y}$ is the observation and $\mathbf{h}$ is the unknown to be estimated. For example, the maximum likelihood (ML) estimate is [1]

$$\hat{\mathbf{h}} = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{y}, \tag{1.2}$$

where $(\cdot)^H$ represents the Hermitian (conjugate-transpose) operator. An example channel measured at Edwards AFB is illustrated in Figure 1.2. Note that the example channel is *sparse*, something that is true of most channels in wireless communications [2]. A channel is considered sparse if most of the coefficients are zero or nearly zero. The ML estimate (1.2) for an SNR of 50 dB is illustrated in Figure 1.3. As expected, the ML estimate and the true channel show close agreement. Of particular interest is the fact that the (time-domain) ML estimate is *sparse*. However, the ML estimate at a more routinely encountered SNR, say 16 dB, is illustrated in Figure 1.4. Here, the (time-domain) ML estimate is *not sparse*, even though there is relatively close agreement between the ML estimate and the true channel in the frequency domain for frequencies corresponding to the 99.5% power bandwidth of the modulated waveform (see the Appendix).

The previous example illustrated the fact that the ML estimate does not always produce a sparse estimate of a sparse vector. This observation compels one to ask if constraining the estimation problem to produce a sparse channel estimate improves equalizer performance. This is the central question of this thesis. The post-equalizer bit error rate (BER) is used to answer this question: if the post-equalizer BER using a sparse channel estimate is lower than the post-equalizer BER using the ML channel estimate, then we say that the performance of the data-aided equalizer has improved as a result of using a sparse channel estimate.

1.4 Organization

This thesis is organized as follows. Chapter 2 starts with a brief section on the notation used throughout this thesis. It also contains the problem formulation as well as a brief discussion on

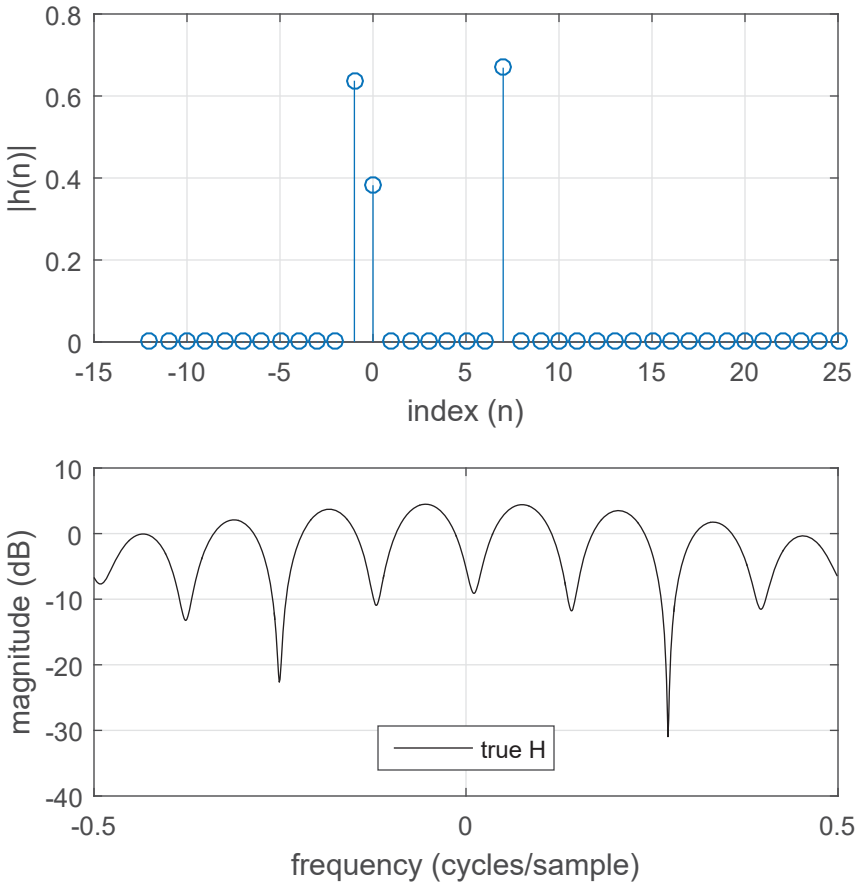


Figure 1.2: An example channel: (top) the impulse response; (bottom) the corresponding frequency domain transfer function.

previous work done estimating sparse signals. Chapter 3 presents the estimators examined in this thesis. Chapter 4 summarize simulation results corresponding to the channel estimators described in Chapter 3. The channels used in Chapter 4 were measured at Edwards Air Force Base. The results show that the General Orthogonal Matching Pursuit has the greatest positive impact on the post-equalizer BER. The next best improvement is achieved by the SPARSEVA-RE, Least Absolute Shrinkage Selector Operator (LASSO), and the Subspace Pursuit (SP). Finally, Chapter 5 is the conclusion and suggestions for future work.

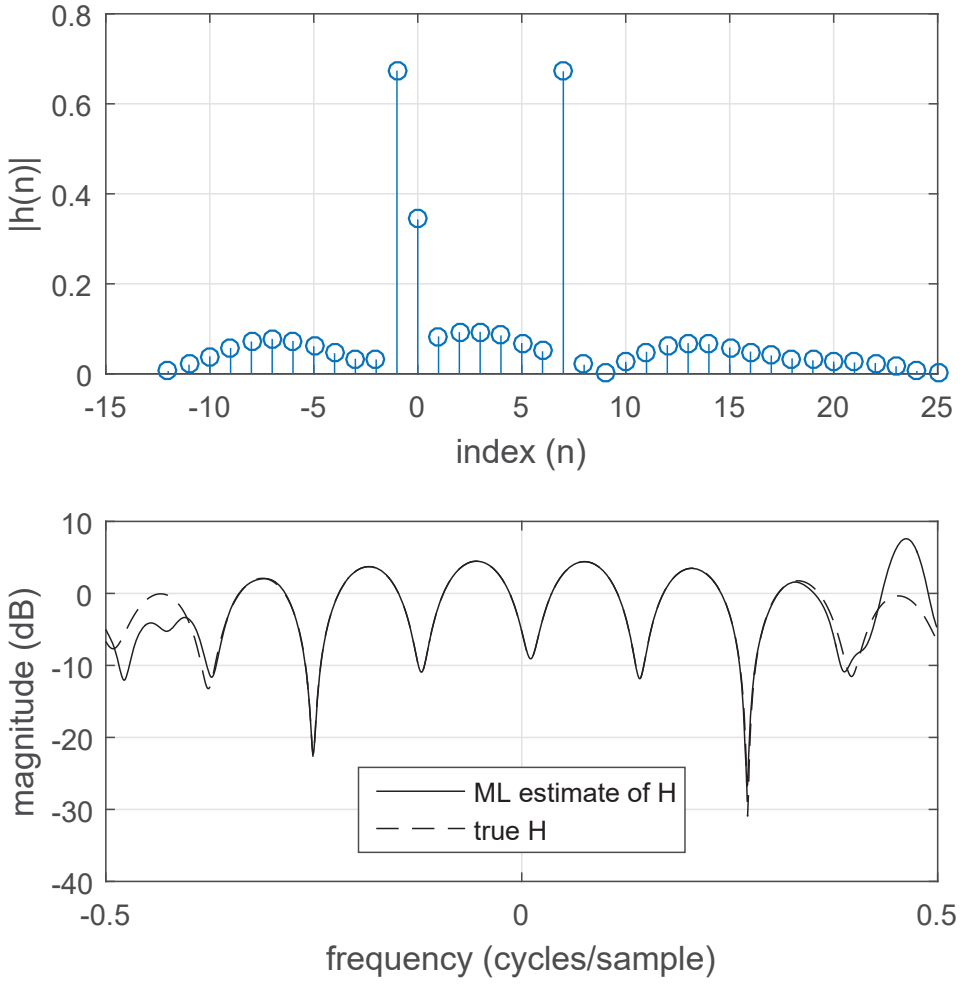


Figure 1.3: ML estimate of the channel at an SNR of 50 dB (top) and corresponding frequency transfer function with the true transfer function (bottom).

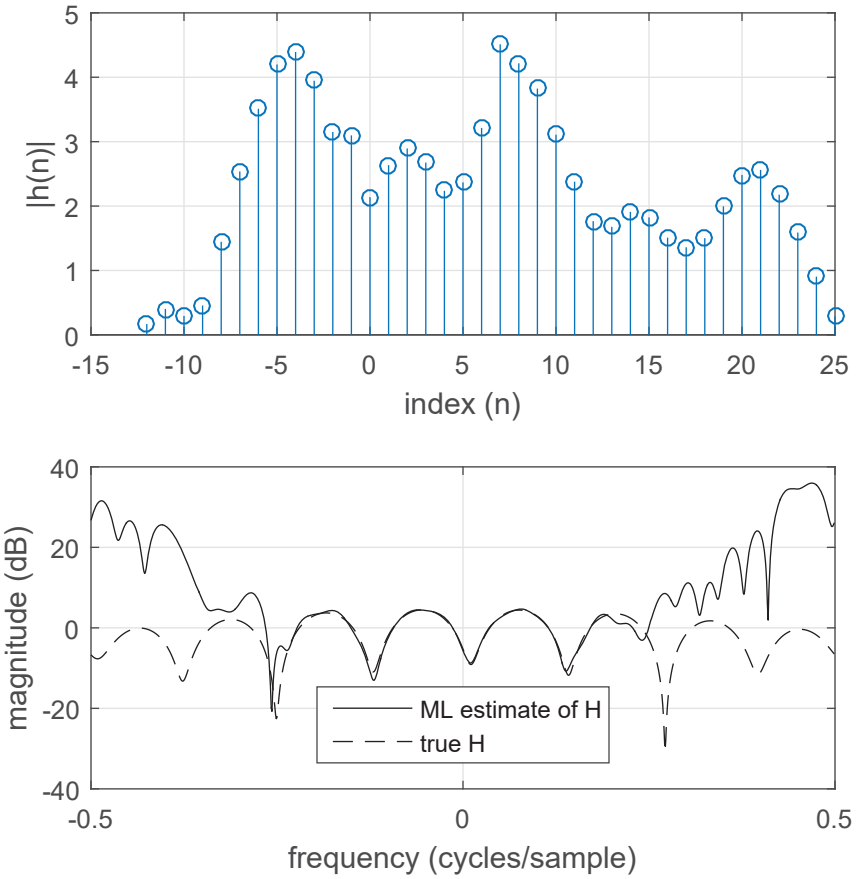


Figure 1.4: ML estimate of the channel at an SNR of 16 dB (top) and corresponding frequency transfer function with the true transfer function (bottom).

CHAPTER 2. LITERATURE REVIEW

2.1 Mathematical Notation

Scalars or constants are denoted with lower case letters, e.g., a. The n-th sample of a sequence of T-spaced samples of a continuous time waveform $s(t)$ is denoted as $s(nT)$. Column vectors and matrices are denoted using lower-case boldface letters, e.g., $\mathbf{a}$ and $\mathbf{A}$, respectively. $\mathbf{I}_N$ denotes the $N \times N$ identity matrix. The n-th element of a vector is denoted $\mathbf{a}_n$ and the element in the n-th row and m-th column of a matrix is denoted $\mathbf{A}_{m,n}$. The complex-conjugate of a scalar, vector, or matrix are denoted a^*, $\mathbf{a}^*$, and $\mathbf{A}^*$, respectively. Transpose and conjugate transpose (Hermitian) of a vector or matrix are denoted $\mathbf{a}^T$, $\mathbf{a}^H$, $\mathbf{A}^T$, or $\mathbf{A}^H$, respectively. The L_p norm of a vector or matrix are denoted $\|\mathbf{a}\|_p$ or $\|\mathbf{A}\|_p$, respectively. An estimate of a scalar, vector, or matrix are denoted using "hats": $\hat{a}$, $\hat{\mathbf{a}}$, or $\hat{\mathbf{A}}$, respectively. Calligraphic letters are used to denote sets, e.g., $\mathcal{T}$. The cardinality of $\mathcal{T}$ (the number of elements in the set $\mathcal{T}$) is denoted $|\mathcal{T}|$. $\hat{\mathcal{T}}$ is used to denote an estimate of the set members. $\mathbf{A}_{\mathcal{T}}$ denotes a sub-matrix of the matrix $\mathbf{A}$ using the columns of $\mathbf{A}$ indexed by the members of the set $\mathcal{T}$. For example, suppose the matrix $\mathbf{A}$ is defined as

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 3 & 5 \\ 1 & 4 & 3 & 6 \\ 1 & 5 & 3 & 7 \\ 1 & 6 & 3 & 8 \end{bmatrix}$$

and $\mathcal{T} = \{1, 3\}$. Then the matrix $\mathbf{A}_{\mathcal{T}}$ is defined as

$$\mathbf{A}_{\mathcal{T}} = \begin{bmatrix} 1 & 3 \\ 1 & 3 \\ 1 & 3 \\ 1 & 3 \\ 1 & 3 \end{bmatrix}.$$

2.2 Problem Formulation

Figure 2.1 (a) is a block diagram representation of the problem. The complex-valued baseband equivalent [3] for all signals is used. The modulator creates the continuous-time SOQPSK-TG signal $s_\ell(t)$, which passes through a multipath channel modeled by an LTI system with impulse response $h_{\ell c}(t)$. Noise is added to the channel output to form the received signal $r_\ell(t)$. The noise, $w(t)$, is modeled as a wide sense stationary complex-valued circularly symmetric normal random process with zero mean and power spectral density $S_w(f) = 2N_0$ W/Hz. In preparation for sampling, an anti-aliasing filter with impulse response $h_{\text{lpf}}(t)$ is applied. For the anti-aliasing filter, we assume an ideal lowpass filter with transfer function shown in Figure 2.2. The output of the anti-aliasing filter is

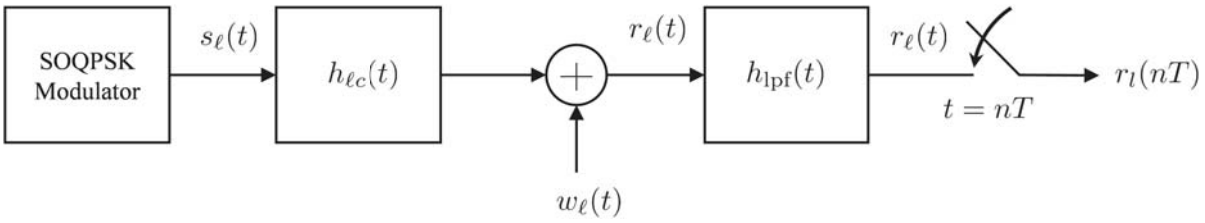
$$r_\ell(t) = s_\ell(t) * \underbrace{h_{\ell c}(t) * h_{\text{lpf}}(t)}_{h(t)} + \underbrace{w_\ell(t) * h_{\text{lpf}}(t)}_{z(t)} \tag{2.1}$$

$$= s_\ell(t) * h(t) + z(t), \tag{2.2}$$

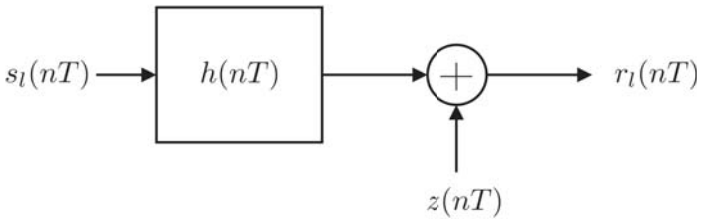
where $z(t)$ is a complex-valued circularly symmetric normal random process with zero mean and autocorrelation function

$$E\left[z(t+\tau)z^*(t)\right] = \frac{2N_0}{T}\text{sinc}\Big(\frac{\tau}{T}\Big), \tag{2.3}$$

where $\text{sinc}\big(x\big) = \sin(\pi x)/\pi x$. The anti-aliasing filter output is sampled at T-spaced intervals to produce the discrete-time sequence $r_\ell(nT)$. The sequence of received samples may be expressed



(a)



(b)

Figure 2.1: A block diagram of the system: (a) the block diagram in terms of the complex-valued lowpass equivalent signals; (b) the equivalent discrete-time system.

as

$$r_\ell(nT) = s_\ell(nT) * h(nT) + z(nT), \tag{2.4}$$

where

$$h(nT) = h(t)\Big|_{t=nT} \tag{2.5}$$

and the $z(nT)$ form a complex-valued circularly symmetric normal random sequence with zero mean and autocorrelation

$$E\left[z((n+m)T)z^*(nT)\right] = \frac{2N_0}{T}\delta(m). \tag{2.6}$$

Equation (2.4) defines the equivalent discrete-time system shown in Figure 2.1 (b).

For the following we use a normalized sample rate corresponding to 2 samples/bit, because this sample rate is the lowest integer multiple of the bit rate that satisfies the sampling theorem [4] for SOQPSK-TG (see the Appendix). The channel contains N_1 non-causal samples and N_2 casual samples. Therefore, the channel is represented as an FIR filter of length $N_1 + N_2 + 1$.

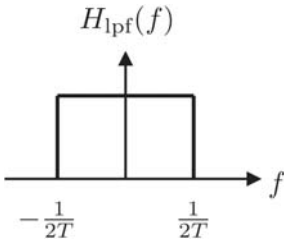


Figure 2.2: The transfer function of the ideal lowpass filter used as the anti-aliasing filter.

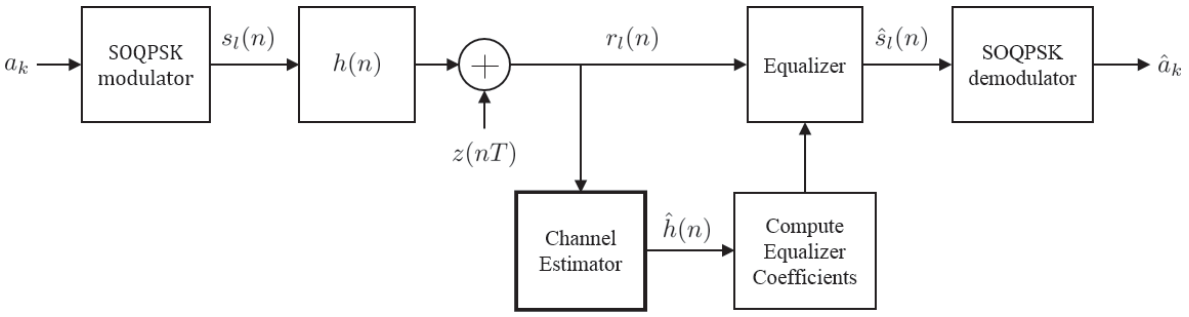


Figure 2.3: Simplified block diagram of problem.

For notational convenience, the N_p waveform samples corresponding to the pilot sequence are $s(0) \ldots s((N_p - 1)T)$. We also assume that $s(mT)$, for $m < 0$ and $m \geq N_p$ represent waveform samples corresponding to unknown data preceding or following the pilot sequence, respectively.

The channel estimator forms the channel estimate using the waveform samples corresponding to the pilot sequence. Let $r_\ell(iT)$ be the received sample corresponding to the start of the pilot. Then, ignoring noise for the moment,

$$\begin{aligned} r_\ell(iT) &= \sum_{m=-N_1}^{N_2} s_\ell(-mT)h(mT) \\ &= h(-N_1T)s_\ell(N_1T) + \cdots + h(0)s_\ell(0) + \cdots + h(N_2T)s_\ell(-N_2T). \end{aligned} \tag{2.7}$$

The first sample that does not depend on the samples corresponding to unknown data preceding the pilot sequence is

$$r_\ell((i+N_2)T) = \sum_{m=-N_1}^{N_2} s((N_2-m)T)h(mT) \tag{2.8}$$

$$= h(-N_1T)s_\ell((N_2+N_1)T) + \cdots + h(N_2T)s_\ell(0). \tag{2.9}$$

The last sample that does not depend on the unknown data following the pilot sequence is

$$r_\ell((i+N_p-N_1-1)T) = \sum_{m=-N_1}^{N_2} s((N_p-N_1-1-m)T)h(mT) \tag{2.10}$$

$$= h(-N_1T)s_\ell((N_p-1)T) + \cdots + h(N_2T)s_\ell(N_p-N_1-N_2-1). \tag{2.11}$$

Two important observations must be made at this point.

1. Equations (2.9) and (2.11) define $N_p - N_1 - N_2$ usable equations for channel estimation. Thus $N_p > N_1 + N_2$ is required for this to work.

2. For $N_p > N_1 + N_2$, we have a choice of how many of the $N_p - N_1 - N_2$ equations are used for channel estimation. In other words, q of the equations may be used to estimate the $Q = N_2 + N_1 + 1$ channel coefficients. Using $q < Q$ equations produces an underdetermined system which has an infinite number of solutions [5]. Using $q = Q$ equations produces a square system with a single solution, as long as the q equations are linearly independent [5]. Using $q > Q$ equations produces an overdetermined system for which there is no solution [5]. In this case, a solution is defined as the vector that minimizes some norm.

In the application considered in this thesis, $N_1 = 12$, $N_2 = 25$, and $N_p = 384$ samples [6]. Consequently, there are $384 - 25 - 12 = 347$ usable equations defined by (2.9) and (2.11). The second point requires some discussion. First, for $q < Q$, the underdetermined system models

the case where a signal has been undersampled or a lower dimensional representation of a discrete data set is desired. In the sparse estimation literature, this problem is usually referred to as *compressed sensing* [7, 8]. Of the infinite number of solutions to the underdetermined system of equations, compressed sensing seeks the sparsest solution, i.e., the one with the fewest non-zero entries. Second, for $q = Q$, because there is only one solution (for a full-rank system), there is no opportunity to impose a sparse constraint on the solution. Third, for $q > Q$, the overdetermined system possess no exact solutions. *Sparse estimation* seeks the sparsest solution that minimizes a given norm. Dai and Pelckmans [9] prove that as the number of equations brought to bear on the estimatE increases, the channel estimate improves. Thus we choose to use all possible equations and formulate the problem in the context of sparse estimation.

Using all the $N_p - N_1 - N_2$ available equations (2.9) and (2.11), the system of equations defining the estimation problem is

$$\begin{bmatrix} r_l((i+N_2)T) \\ \vdots \\ r_l((i+N_p-N_1-1)T) \end{bmatrix} = \begin{bmatrix} s((N_2+N_1)T) & \cdots & s(0) \\ \vdots & & \vdots \\ s((N_p-1)T) & \cdots & s((N_p-N_1-N_2-1)T) \end{bmatrix} \begin{bmatrix} h(-N_1T) \\ \vdots \\ h(N_2T) \end{bmatrix} + \begin{bmatrix} z((i+N_2+N_1)T) \\ \vdots \\ z((i+N_p-1)T) \end{bmatrix} \quad . \quad (2.12)$$

Equation (2.12) can be written as

$$\mathbf{y} = \mathbf{A}\mathbf{h} + \mathbf{z}, \quad (2.13)$$

where $\mathbf{y}$ is the $q \times 1$ [$q = (N_p - N_2 - N_1)$] vector of received samples, $\mathbf{A}$ is the $q \times Q$ [$Q = (N_2 + N_1)$] convolution matrix formed from samples of the waveform corresponding to the pilot

sequence, $\mathbf{h}$ is the $Q \times 1$ vector of channel coefficients, and $\mathbf{z}$ is the $q \times 1$ noise vector. In the sparse channel estimation literature, the known matrix $\mathbf{A}$ is commonly referred to as the *sensing matrix*.

2.3 Review

Finding sparse solutions to (2.13) has been well studied: see [2, 7–21] and the references therein. Some of the algorithms developed in these references assume an underdetermined system, however, they make no claim that the algorithms *only* work for underdetermined systems. Therefore, we apply the algorithms to our overdetermined case.

There are three categories of algorithms that produce sparse estimates, optimization algorithms, greedy pursuit algorithms, and thresholding algorithms. Optimization algorithms typically create sparse estimates in one of two ways. The first minimizes the L_2 norm using the L_1 norm as a constraint. An example of this type of optimization algorithm is the least absolute shrinkage and selection operator (LASSO) introduced by Tibshirani [11]:

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{A}\mathbf{h}\|_2^2 \quad \text{subject to} \quad \|\mathbf{h}\|_1 \leq t. \tag{2.14}$$

Later, Zou [15] sought to improve LASSO by adding a weighting vector to the constraint to produce an algorithm he called the Adaptive LASSO (AdaLASSO):

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{A}\mathbf{h}\|_2^2 \quad \text{subject to} \quad \mathbf{w}^T|\mathbf{h}| \leq t, \tag{2.15}$$

where $\mathbf{w}$ is the $q \times 1$ vector of weights based on the sample size.

The second approach is to minimize the L_1 norm subject to a constraint. An algorithm that follows this pattern was introduced by Dai and Pelckmans [9]. Starting with the ML estimate $\hat{\mathbf{h}}^{ML}$

given by (1.2), the sparse estimate is produced in four steps:

$$\begin{aligned}
\hat{\mathbf{h}}^{LP} &= \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{h}\|_1 \quad \text{subject to} \quad \|\mathbf{h} - \hat{\mathbf{h}}^{ML}\|_\infty \leq \lambda \\
\mathcal{T} &= \text{indexes of nonzero elements in } \hat{\mathbf{h}}^{LP} \\
\hat{\mathbf{h}}_{\mathcal{T}} &= \left(\mathbf{A}_{\mathcal{T}}^H \mathbf{A}_{\mathcal{T}}\right)^{-1} \mathbf{A}_{\mathcal{T}}^H \mathbf{y} \\
\hat{\mathbf{h}} &= \text{sparse_expand}\left(\hat{\mathbf{h}}_{\mathcal{T}}, \mathcal{T}, Q\right),
\end{aligned} \tag{2.16}$$

where $\mathbf{y} = \text{sparse\_expand}(\mathbf{x}, \mathcal{T}, Q)$ denotes the creation of a $Q \times 1$ vector $\mathbf{y}$ from a $|\mathcal{T}| \times 1$ vector ($|\mathcal{T}| < Q$) $\mathbf{x}$ as follows. Let $\mathcal{T} = \{n_1, n_2, \ldots, n_{|\mathcal{T}|}\}$. Then

$$y_n = \begin{cases} x_i & n = n_i \in \mathcal{T} \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

By adjusting the threshold λ, the algorithm keeps the largest (in magnitude) elements of $\hat{\mathbf{h}}^{ML}$ and replaces the other elements of $\hat{\mathbf{h}}^{ML}$ with zero. Others like the Dantzig Selector [10] minimize the L_1 norm of the estimate subject to a constraint on the projection of the residual vector onto each of the columns of the sensing matrix:

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{h}\|_1 \quad \text{such that} \quad \|\mathbf{A}^H(\mathbf{y} - \mathbf{A}\mathbf{h})\|_\infty \leq t. \tag{2.18}$$

The greater the error is allowed to be (i.e., the larger t is), the more sparse the estimate is.

The second category, the greedy pursuits, are iterative and seek to find the columns in the sensing matrix that have the greatest impact on the received vector at each iteration. Most find the columns by projecting the residual vector onto each of the columns of $\mathbf{A}$, computing the squared scalar projection, and selecting the columns corresponding to the largest squared scalar projection. The selected columns are then used to update the estimate and the residual vector and the process is repeated until some stopping criterion is met. Mallat and Zhang [20] introduced an algorithm called the Matching Pursuit that showed the effectiveness of such methods in producing sparse

estimates. The Matching Pursuit algorithm sometimes selects the same column of $\mathbf{A}$ twice and re-estimate the corresponding coefficient. In an effort to avoid selecting the same column twice, Pati, Rezaiifar, and Krishnaprasad [21] developed the Orthogonal Matching Pursuit. The Orthogonal Matching Pursuit guarantees that the selection of a column only happens once.

The third category comprises thresholding algorithms. Thresholding algorithms are very similar to the greedy pursuit algorithms. They are iterative and typically select the best column of $\mathbf{A}$ using the same method as greedy pursuits. The biggest difference is that thresholding algorithms typically estimate more than the required number of coefficients and then have a pruning step to replace the unnecessary coefficients with zeros. The Subspace Pursuit, introduced by Dai and Milenkovic [13], estimates up to twice the number of desired coefficients and then keeps the largest k while setting the rest to zero.

In summary, there are a number of sparse estimation techniques based on a number of different criterion. The open question is how well these work in the aeronautical telemetry setting.

CHAPTER 3. SPARSE ESTIMATION ALGORITHMS FOR SPARSE CHANNEL RECONSTRUCTION

3.1 Algorithms

3.1.1 Restricted Isometry Property (RIP)

Some of the sparse estimation algorithms rely on the restricted isometry property (RIP) to guarantee recovery of the sparse channel. The matrix $\mathbf{A}$ satisfies RIP of order, k if there exists a $\delta_k \in (0, 1)$ such that

$$(1 - \delta_k)||\mathbf{h}||_2^2 \leq ||\mathbf{A}\mathbf{h}||_2^2 \leq (1 + \delta_k)||\mathbf{h}||_2^2 \tag{3.1}$$

holds for all $\mathbf{h} \in \{\mathbf{h} : ||\mathbf{h}||_0 < k\}$ [22]. If the matrix $\mathbf{A}$ does not satisfy RIP, then there is no guarantee that the sparse channel will be recovered. However, calculating RIP is difficult for large matrices. In practice, most sensing matrices $\mathbf{A}$ are formed from a Gaussian random process as they satisfy RIP with high probability [23]. In our application, the sensing matrix $\mathbf{A}$ is not random, but constructed as outlined in Section 2.2. It is difficult to tell if the matrix $\mathbf{A}$ defined in (2.12) and (2.13) satisfies the RIP. We make the assumption that $\mathbf{A}$ satisfies RIP, knowing that if it does not satisfy RIP, some of the algorithms may perform poorly.

3.1.2 Maximum Likelihood

Here we look at a brief derivation of the maximum likelihood estimate and, at the end, will see it is equivalent to the least squares estimate. The maximum likelihood estimate is [1]

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmax}}\, f_{\mathbf{h}}(\mathbf{y}), \tag{3.2}$$

where $f_{\mathbf{h}}(\mathbf{y})$ is the probability density function (PDF) of $\mathbf{y}$ given $\mathbf{h}$. Substituting the PDF of $\mathbf{y}$ into (3.2)

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmax}}[\frac{1}{\pi^N \sigma^{2N}} e^{\frac{1}{\sigma^2}(\mathbf{y}-\mathbf{Ah})^H(\mathbf{y}-\mathbf{Ah})}] \tag{3.3}$$

$$= \underset{\mathbf{h}}{\operatorname{argmax}}[-N \ln(2\pi\sigma^2) - \frac{1}{\sigma^2}(\mathbf{y}-\mathbf{Ah})^H(\mathbf{y}-\mathbf{Ah})] \tag{3.4}$$

$$= \underset{\mathbf{h}}{\operatorname{argmax}}[-(\mathbf{y}-\mathbf{Ah})^H(\mathbf{y}-\mathbf{Ah})] \tag{3.5}$$

$$= \underset{\mathbf{h}}{\operatorname{argmax}}[-(\mathbf{y}\mathbf{y}^H - \mathbf{y}^H\mathbf{Ah} - \mathbf{h}^H\mathbf{A}^H\mathbf{y} + \mathbf{h}^H\mathbf{A}^H\mathbf{Ah})]. \tag{3.6}$$

The step from (3.3) to (3.4) is permissible, because the natural log is a monotonic function and does not change which value of $\mathbf{h}$ maximizes the function. Going from (3.4) to (3.5), the constants and scalars are dropped for the same reason. We can now take the derivative of (3.6) with respect to $\mathbf{h}^H$, set it equal to zero, and solve for $\hat{\mathbf{h}}$ as follows:

$$\frac{\partial}{\partial \mathbf{h}^H}\hat{\mathbf{h}} = \frac{\partial}{\partial \mathbf{h}^H}[-(\mathbf{y}\mathbf{y}^H - \mathbf{y}^H\mathbf{Ah} - \mathbf{h}^H\mathbf{A}^H\mathbf{y} + \mathbf{h}^H\mathbf{A}^H\mathbf{Ah})] = \mathbf{0}$$

$$\mathbf{A}^H\mathbf{A}\hat{\mathbf{h}} = \mathbf{A}^H\mathbf{y}$$

$$\hat{\mathbf{h}} = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{y}, \tag{3.7}$$

where $(\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H$ is the Moore Penrose pseudoinverse. The least squares estimate is given by

$$\mathbf{h}^{LS} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{y}-\mathbf{Ah}\|_2^2$$

$$= \underset{\mathbf{h}}{\operatorname{argmin}} (\mathbf{y}-\mathbf{Ah})^H(\mathbf{y}-\mathbf{Ah})$$

$$= \underset{\mathbf{h}}{\operatorname{argmax}}[-(\mathbf{y}-\mathbf{Ah})^H(\mathbf{y}-\mathbf{Ah})]. \tag{3.8}$$

Note that (3.8) is equal to (3.5). This is true because the noise is white. An example ML estimate's performance is given in Section 1.3.

3.1.3 Optimization Algorithms

Brute Force

As shown previously, the ML estimate does not take into consideration the known sparsity. By forcing the estimates to be sparse, it is possible to more accurately model the channel. One way to force a sparse estimate is to perform the ML estimate of the k non-zero locations. We let $k = 6$ for our application, because the number of non-zero coefficients the channels measured at Edwards AFB on average is six. The estimate is computed by

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{A}\mathbf{h}\|_2^2 \qquad \text{such that} \qquad \|\mathbf{h}\|_0 = k. \tag{3.9}$$

This requires trying all $\binom{n}{k} = \binom{38}{6} = 2,760,681$ possible combinations of $\mathcal{T}$. In our experiment, $n = 38$ resulting in 2,760,681 combinations! Performing so many pseudoinverses requires a significant amount of time, making it unrealistic to use in a real time system. Implementing this approach in MATLAB, the time required to solve (3.9) is over 12 minutes. Many of the algorithms explored later in this thesis choose which of the columns of $\mathbf{A}$ work well in building the estimate $\hat{\mathbf{h}}$, without having to enumerate all of the possible combinations.

Figures 3.1 and 3.2 are the time-domain estimate and frequency response respectively. The time-domain estimate and frequency response are nearly identical to the true channel at SNR 16 dB. This algorithm does an excellent job of estimating the true channel.

Least Absolute Shrinkage and Selection Operator (LASSO)

The Least Absolute Shrinkage and Selection Operator (LASSO) was first introduced in [11] and draws on principles from ridge regression and subset selection. The LASSO algorithm shrinks some coefficients while setting others equal to zero, which allows LASSO to create sparse estimates. Specifically, LASSO solves the minimization problem

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{A}\mathbf{h}\|_2^2 \quad \text{subject to} \quad \|\mathbf{h}\|_1 \leq t, \tag{3.10}$$

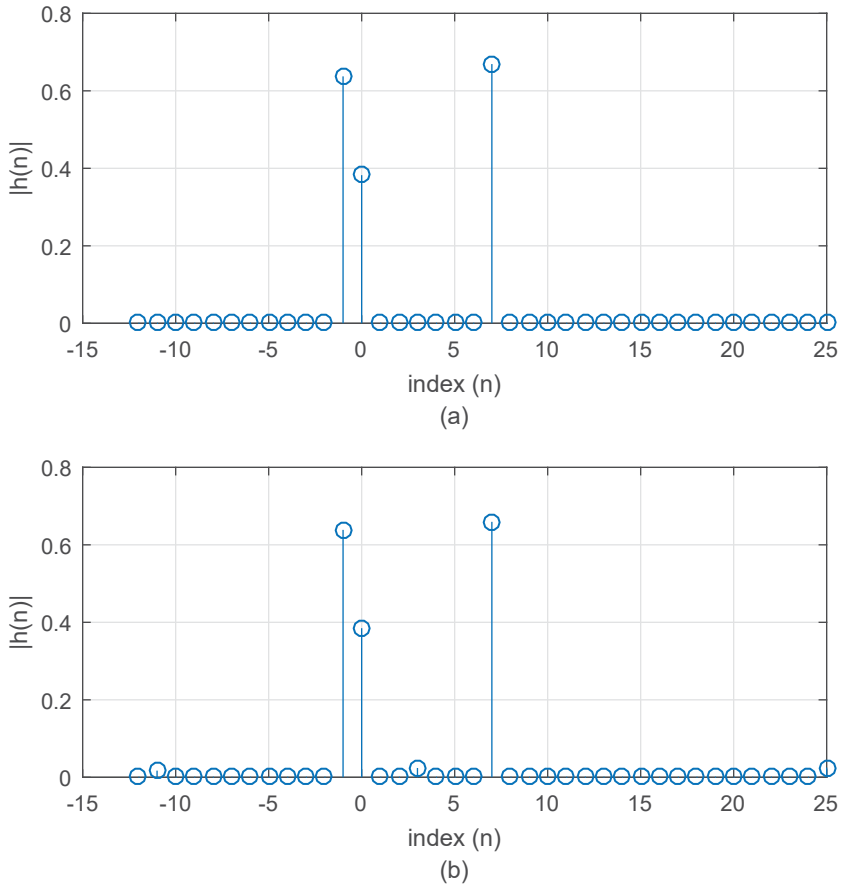


Figure 3.1: True channel measured on Taxiway E; (b) the brute force channel estimate at SNR = 16 dB.

where t is considered to be the tuning parameter. As t gets smaller, the number of zero coefficients in the estimate increases. It is more common to see the Lagrangian form of Equation (3.10)

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{h}\|_2^2 + \lambda\|\mathbf{h}\|_1, \tag{3.11}$$

where λ is the new tuning parameter and, as λ increases, the coefficients are shrunk until they are forced to zero. Equations (3.10) and (3.11) are equivalent, because for a given t there exists a single λ that produces the same result. There is no closed form solution for Equation (3.11), so finding a solution is a quadratic programming problem [24].

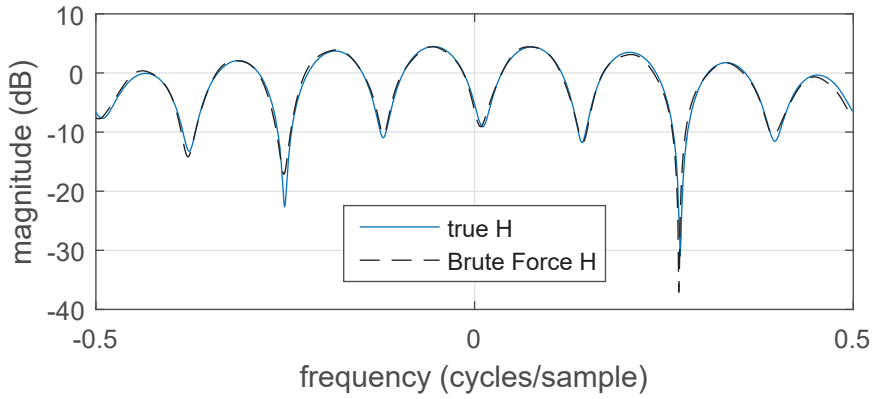


Figure 3.2: The frequency response of the true channel and the brute force estimate at SNR = 16 dB.

The MATLAB Statistics and Machine Learning Toolbox includes a function called lasso. Here we use the lower case "lasso" to denote the MATLAB function and the all capital "LASSO" to denote the general technique. Table 3.1 contains the code for the implementation of the lasso function. For this application, lasso requires four arguments: the sensing, the vector of received samples, and a pair of arguments instructing the routine to find the k best non-zero values. Note that MATLAB's lasso function operates only on real-valued data, so that matrices and vectors formed by stacking their real and imaginary parts are passed as arguments. The output of LASSO is a matrix of real elements whose columns are estimates. The first column corresponds to the estimate with the smallest λ that produces twelve non-zero coefficients, which is the estimate we want. We estimate 12 coefficients, the real and imaginary parts of six complex-valued coefficients.

Table 3.1: LASSO

```
function h_hat = LASSO_CS_ChannelEstimate(L, y, A, k)
        y = [real(y); imag(y)];
        phi = [real(A), -imag(A); imag(A), real(A)];
        output = lasso(phi,y,'DFmax',k);
        h_hat = output(1:38,1)+1i*output(39:end,1);
end
```

Figure 3.3 shows the true channel and the LASSO estimate at SNR = 16 dB. The LASSO function does a remarkable job recovering the signal, doing so almost exactly. We should expect the frequency response to be nearly identical to that of the true channel and Figure 3.4 demonstrates this.

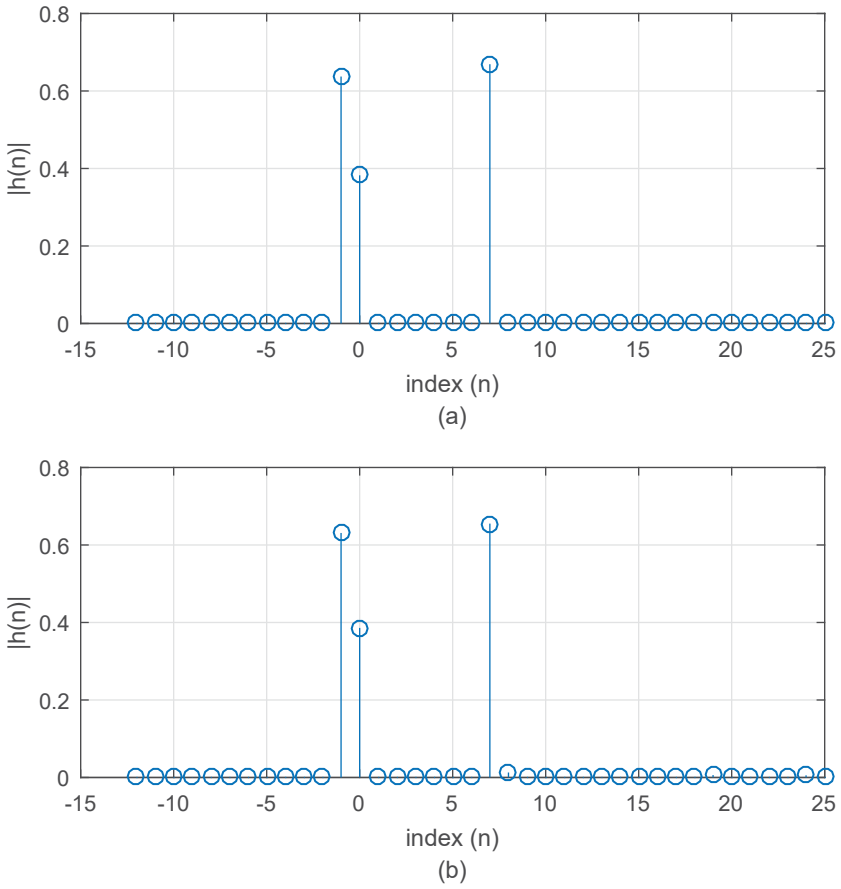


Figure 3.3: True channel measured on Taxiway E; (b) the LASSO channel estimate at SNR = 16 dB.

Adaptive LASSO

The Adaptive LASSO (AdaLASSO) is a slight variation of the LASSO algorithm [15]. Instead of equally penalizing each of the coefficients in the L_1 norm, a weighting vector is introduced, changing the penalty on a per coefficient basis. Let $\tilde{\mathbf{h}}$ be an estimate of $\mathbf{h}$ and choose a $\gamma > 0$. The

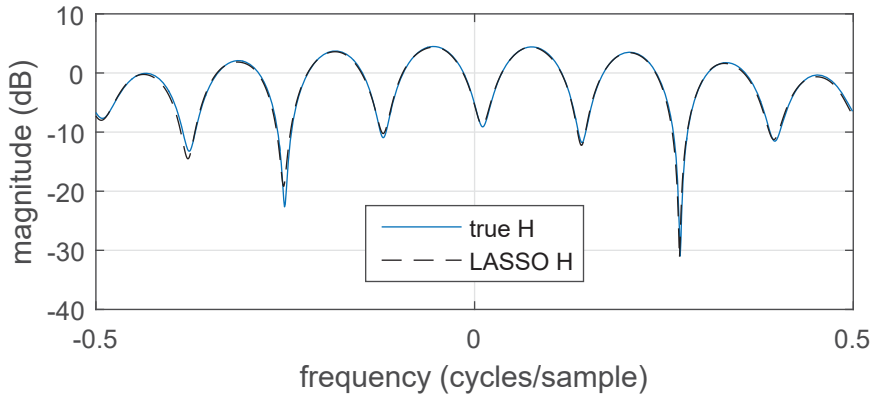


Figure 3.4: The frequency response of the true channel and the LASSO estimate at SNR = 16 dB.

weight vector is defined as $\mathbf{w} = 1/\tilde{\mathbf{h}}^{\gamma}$. The resulting equation is

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{A}\mathbf{h}\|_2^2 + \lambda \sum_{j=0}^{N1+N2-1} w_j |h_j|. \tag{3.12}$$

A commonly used estimate is the ML estimate; $\tilde{h}$ is the ML estimate in the results below. We let $\gamma = 0.01$ and $\lambda = \sqrt{Q}$, which is the number of columns in $\mathbf{A}$. As $\lim_{\gamma \to 0}$ AdaLASSO = LASSO.

The MATLAB code for the AdaLASSO algorithm is listed in Table 3.2. Like the lasso function from MATLAB, the fmincon function only accepts real values, necessitating the reconstruction of $\mathbf{A}$ and $\mathbf{y}$. It should be noted that the MATLAB function lasso accepts a paramter limiting the number of coefficients to estimate. There is no such parameter for the AdaLASSO algorithm.

Figure 3.5 is the time domain estimate of the channel at an SNR of 16 dB. The AdaLASSO algorithm was successful in estimating the three non-zero coefficients of the channel. However, AdaLASSO estimated three strong coefficients that should have been zero. The frequency response of the channel estimate is shown in Figure 3.6.

Sparse Estimation based on Validation Re-estimated Least Squares

The Sparse Estimation based on Validation Re-estimated Least Squares (SPARSEVA-RE) algorithm is a three step alogrithim [14] as follows:

Table 3.2: AdaLASSO

```
function h_hat = adalasso(y, A)
        phi = [real(A) -imag(A); imag(A) real(A)];
        h_ols = pinv(A)*y;
        y_r = [real(y);imag(y)];
        h_ols = [real(h_ols); imag(h_ols)];
        gamma = 0.01; %gamma>0
        w = (1./(abs(h_ols).^gamma));
        lambda = sqrt(76);

        fun = @(x)norm(y_r - phi*x,2)^2+lambda*sum(w.*abs(x));
        h0 = ones(size(h_ols));
        phi = [];
        b = [];
        Aeq = [];
        beq = [];
        lb = [];
        ub = [];
        nonlcon = [];
        options = optimoptions('fmincon','MaxFunEvals',3000);
        output = fmincon(fun,h0,pji,b,Aeq,beq,lb,ub,nonlcon,options);

        h_hat = output(1:length(h_hat)/2)+1i*output(1+length(h_hat)/2:end);
end
```

1. $\hat{\mathbf{h}}^{LS} = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{y}$,

2. $\hat{\mathbf{h}}^{LP} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{h}\|_1 \quad \text{subject to} \quad \|(\mathbf{y} - \mathbf{A}\mathbf{h})\|_2^2 \leq \|(\mathbf{y} - \mathbf{A}\hat{\mathbf{h}}^{LS})\|_2^2(1 + \frac{2Q}{q})$, and

3. $\hat{\mathbf{h}} = (\mathbf{A}_{\mathcal{T}}^H\mathbf{A}_{\mathcal{T}})^{-1}\mathbf{A}_{\mathcal{T}}^H\mathbf{y} \quad$ where $\mathcal{T}$ = indexes of non-zero elements of $\hat{\mathbf{h}}^{LP}$.

By allowing the squared error of the linear programming estimate to be less than the scaled squared error of the least squares estimate, the linear programming algorithm finds a sparse solution.

Table 3.3 contains the MATLAB implementation of SPARSEVA-RE. The MATLAB function, fmincon, requires that the objective function be continuous on the first and second derivative. The L_1 norm is not continuous and must be recast as a system of linear equations. Lines 7 through 19 serve this purpose. Figures 3.7 and 3.8 are the time domain estimate using SPARSEVA-RE and its corresponding frequency response at an SNR of 16 dB.

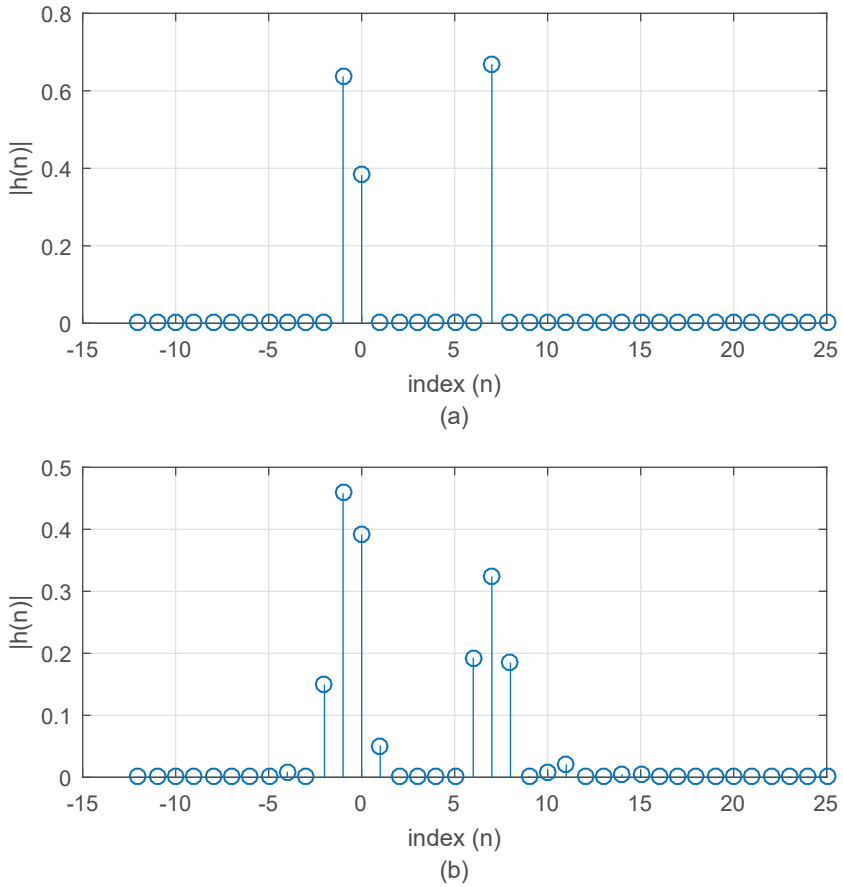


Figure 3.5: True channel measured on Taxiway E; (b) the AdaLASSO channel estimate at SNR = 16 dB.

Dai and Pelckmans Algorithm

Dai and Pelckmans created a three step algorithm, which we call the (DP) algroithm, similar to SPARSEVA-RE [9]. The steps are as follows:

1. $\hat{\mathbf{h}}^{LS} = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{y}$

2. $\hat{\mathbf{h}}^{LP} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{h}\|_1 \quad \text{subject to} \quad \|\mathbf{h} - \hat{\mathbf{h}}^{LS}\|_\infty \leq \lambda$

3. $\hat{\mathbf{h}} = (\mathbf{A}_\mathcal{T}^H\mathbf{A}_\mathcal{T})^{-1}\mathbf{A}_\mathcal{T}^H\mathbf{y} \quad$ where $\mathcal{T} =$ indexes of non-zero elements of $\hat{\mathbf{h}}^{LP}$

where $\lambda = \sqrt{2Q/q^{1-\epsilon}}$ for $0 < \epsilon < 1$.

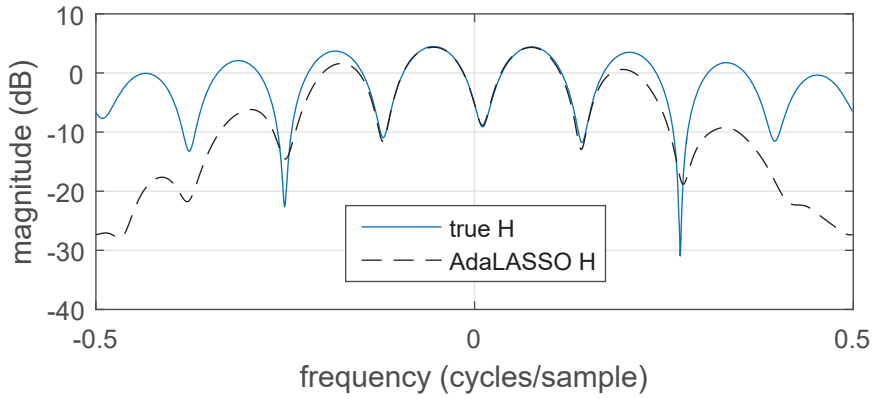


Figure 3.6: The frequency response of the true channel and the AdaLASSO estimate at SNR = 16 dB.

The linear programming equation has an analytic solution given by

$$\hat{\mathbf{h}}_i^{LP} = \begin{cases} 0, & |\mathbf{h}_i^{LS}| \leq \lambda \\ \mathbf{h}_i^{LS} - \lambda, & > \lambda \\ \mathbf{h}_i^{LS} + \lambda, & < -\lambda \end{cases} . \tag{3.13}$$

The DP algorithm preserves the largest coefficients produced by the least squares estimate and sets the rest to zero.

Table 3.4 contains the MATLAB implementation of DP. Just as before, real-valued vectors and matrices have been formed by stacking their real and imaginary parts. This allows the thresholding to work on the real and imaginary coefficients separately. Figures 3.9 and 3.10 are the time domain estimate and its associated frequency response. It is important to note the location of the largest coefficients of the ML estimate vary widely due to noise.

Dantzig Selector

The Dantzig Selector (DS) [2, 10] is the solution to the constrained optimization

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{h}\|_1 \quad \text{such that} \quad \|\mathbf{A}^H(\mathbf{y} - \mathbf{A}\mathbf{h})\|_\infty \leq \sqrt{2\sigma^2(1+a)\log Q} \tag{3.14}$$

Table 3.3: SPARSEVA-RE

```
function h_re = sparseva_re(~, y, A, ~)
        y_r = [real(y);imag(y)];
        phi = [real(A) -imag(A);imag(A) real(A)];
        hls = pinv(A)*y;
        hls_r = [real(hls); imag(hls)];  %least squares reconstructed
        L = length(hls_r);
        phi = zeros(2*L,2*L);

        for idx = 1:L
                phi(idx*2-1,idx)=1;
                phi(idx*2-1,L+idx) = -1;
                phi(idx*2,idx) = -1;
                phi(idx*2,L+idx) = -1;
        end
        b = zeros(2*L,1);
        f = [zeros(L,1);ones(L,1)]';
        fun = @(x)f*x;
        h0 = zeros(2*L,1);
        nonlcon = @(x)constraint(x,hls_r,y_r,phi);
        options = optimoptions('fmincon','Display','off');
        output = fmincon(fun,h0,phi,b,[],[],[],[],nonlcon,options);
        h_lp = output(1:38)+1i*output(39:76);

        %set coefficients that are close to zero to zero
        h_lp(abs(h_lp)<0.05) = 0;
        h_re = zeros(38,1);
        h_re(abs(h_lp)>0) = pinv(A(:,abs(h_lp)>0))*y;

end
function [c,ceq] = constraint(x, hls_r,y_r,phi)
        epsilon = 2*76/(347*2);
        ceq = [];
        c = (y_r-phi*x(1:76))'*(y_r-phi*x(1:76))-(y_r-phi*hls_r)'*...
        (y_r-phi*hls_r)*(1+epsilon);
end
```

for any $a \geq 0$. Q is the number of columns in the sensing matrix $\mathbf{A}$ and σ^2 is the noise variance. The DS seeks to impose sparsity by minimizing the L_1. In our application, we let $a = 1$. Table 3.5 contains the MATLAB code to implement the DS algorithm. The variable `nvar` is the noise variance.

The channel estimate for the DS algorithm at an SNR of 16 dB is shown in Figure 3.11. The DS algorithm is able to find the location of the three largest coefficients. The frequency response

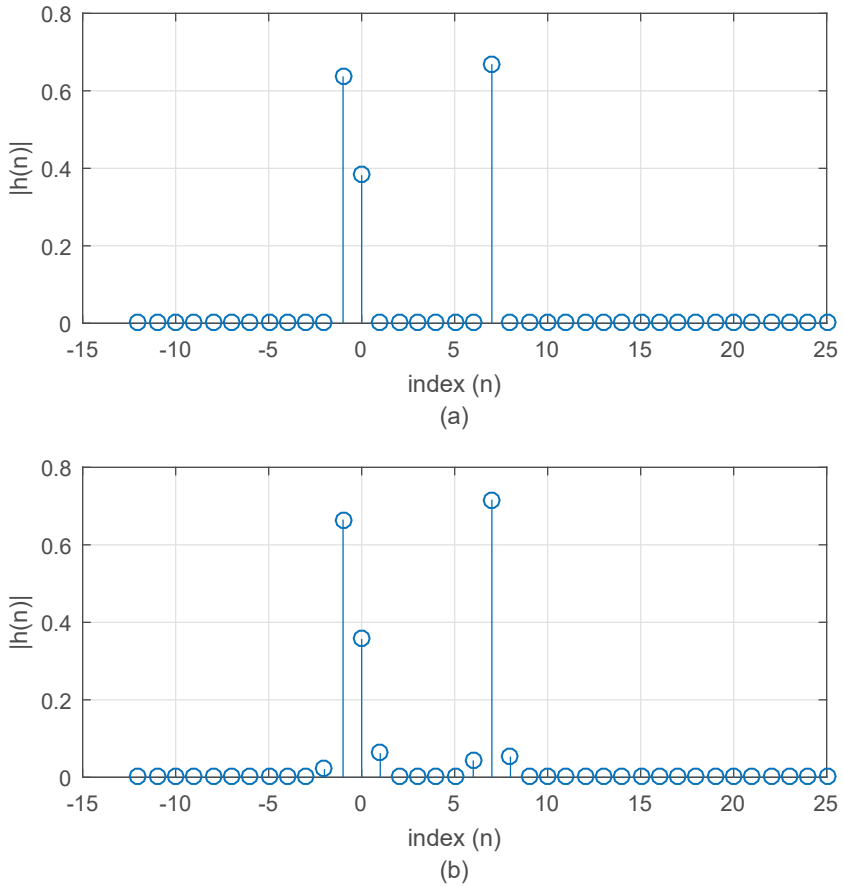


Figure 3.7: True channel measured on Taxiway E; (b) the SPARSEVA-RE channel estimate at SNR = 16 dB.

of the DS estimate is shown in Figure 3.12. The DS estimate is a poor estimate when comparing the frequency response of the true channel to the frequency response of the DS estimate.

Sparse Singular Value Decomposition Estimation

Suppose we wish to obtain a sparse estimate using only the vectors corresponding to non-zero singular values of $\mathbf{A}$. Let $\mathbf{A} = \mathbf{USV}^H$ be the singular value decomposition (SVD) of $\mathbf{A}$, where $\mathbf{U}$ is a $q \times q$ unitary matrix, $\mathbf{S}$ is a $q \times Q$ diagonal matrix consisting of the singular values of $\mathbf{A}$, and

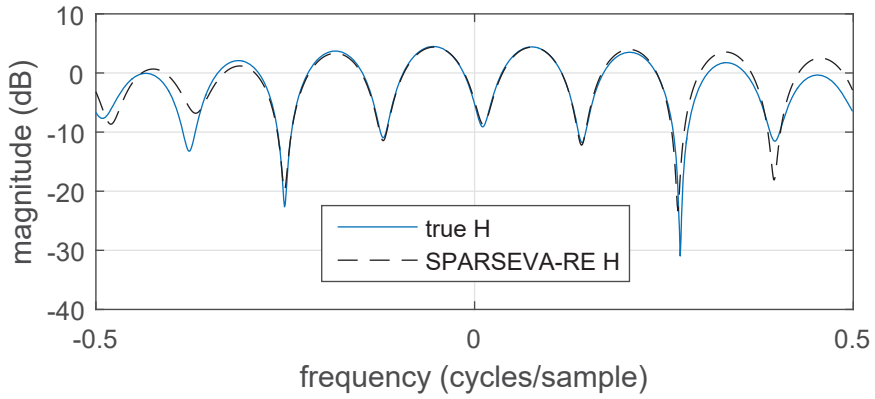


Figure 3.8: The frequency response of the true channel and the SPARSEVA-RE estimate at SNR = 16 dB.

Table 3.4: DP

```
function [ h_hat ] = Dai_pelckmans( L, y, A_con, lambda )
        h_ls = pinv(A_con)*y;
        h_ls_r = [real(h_ls); imag(h_ls)];  %least squares reconstructed

        epsilon = 0.5; % 0<epsilon<1
        lambda = sqrt(2*76/(347*2)^(1-epsilon));

        h_lp_r = h_ls_r;
        h_lp_r(abs(h_ls_r)<lambda) = 0;
        h_lp_r(h_lp_r>0) = h_ls_r(xlp_r>0)-lambda;
        h_lp_r(h_lp_r<0) = h_ls_r(xlp_r<0)+lambda;

        h_lp = h_lp_r(1:38)+1i*h_lp_r(39:76);

        h_re = zeros(38,1);
        h_re(abs(h_lp)>0) = pinv(A_con(:,abs(h_lp)>0))*y;
end
```

$\mathbf{V}$ is an $Q \times Q$ unitary matrix. A unitary matrix has the property that $\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I}$. Let

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_{k\times k} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.15}$$

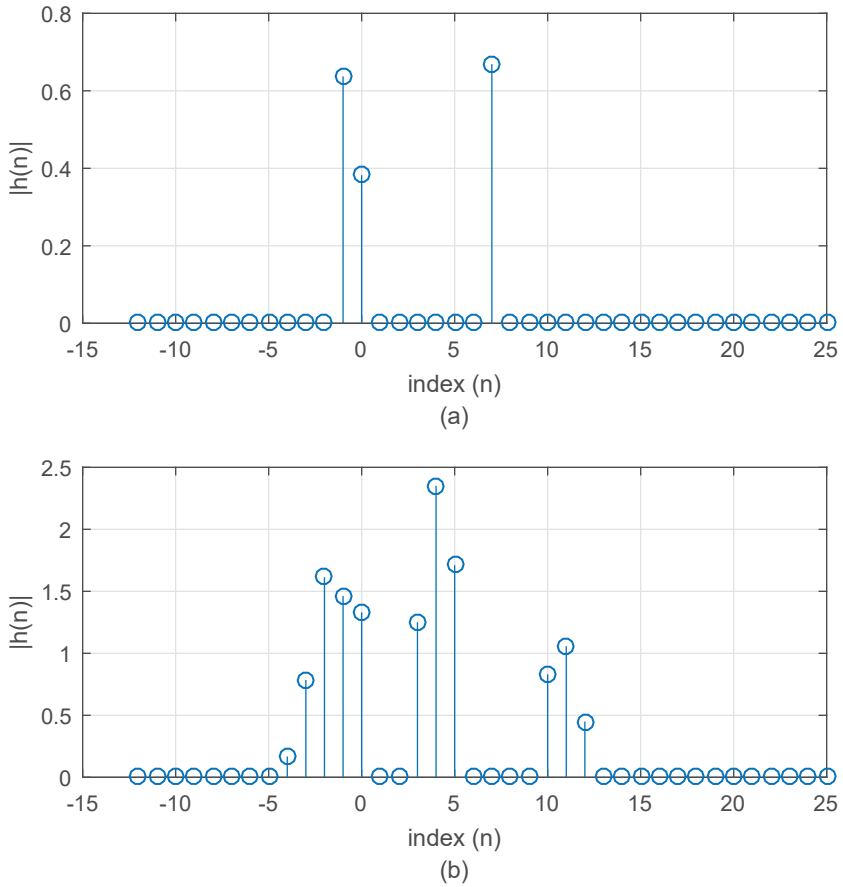


Figure 3.9: True channel measured on Taxiway E; (b) the DP channel estimate at SNR = 16 dB.

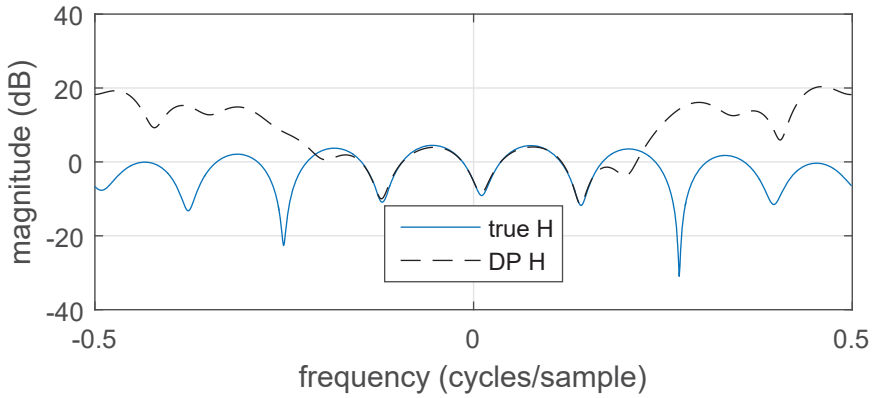


Figure 3.10: The frequency response of the true channel and the DP estimate at SNR = 16 dB.

Table 3.5: DS

```
function [ h_hat ] = DS_CS_ChannelEstimate( L,y,A,k,nvar,hlong )
        %%%Normalize the columns of A
        denoma = sqrt(diag(A'*A));
        A_n = zeros(size(A));
        for idx = 1:length(denoma)
                A_n(:,idx) = A(:,idx)/denoma(idx);
        end

        y_r = [real(y);imag(y)];
        A_r = [real(A_n) -imag(A_n);imag(A_n) real(A_n)];
        a = 1;
        L = L*2;
        phi = zeros(2*L,2*L);
        lambda = sqrt(2*nvar*(1+a)*log10(347*2));

        for idx = 1:L
                phi(idx*2-1,idx)=1;
                phi(idx*2-1,L+idx) = -1;
                phi(idx*2,idx) = -1;
                phi(idx*2,L+idx) = -1;
        end
        b = zeros(2*L,1);
        f = [zeros(L,1);ones(L,1)]';
        fun = @(h)f*h;
        h0 = zeros(2*L,1);
        nonlcon = @(h)dantzig_constraints(h,A_r,y_r,lambda);
        options = optimoptions('fmincon','Display','off');
        output = fmincon(fun,h0,phi,b,[],[],[],[],nonlcon,options);
        h_hat = output(1:38)+1i*output(39:76);
        %normalize estimate
        h_hat = h_hat/(sqrt(h_hat'*h_hat));
end

function [c, ceq] = dantzig_constraints(h,A_r,y_r,lambda)
    c = norm(A_r'*(y_r-A_r*h(1:76)),Inf)-lambda;
    ceq = [];
end
```

be a selection matrix that preserves the information associated with the largest k singular values of $\mathbf{A}$ and sets everything else to zero. Equation (2.13) can be re-written as

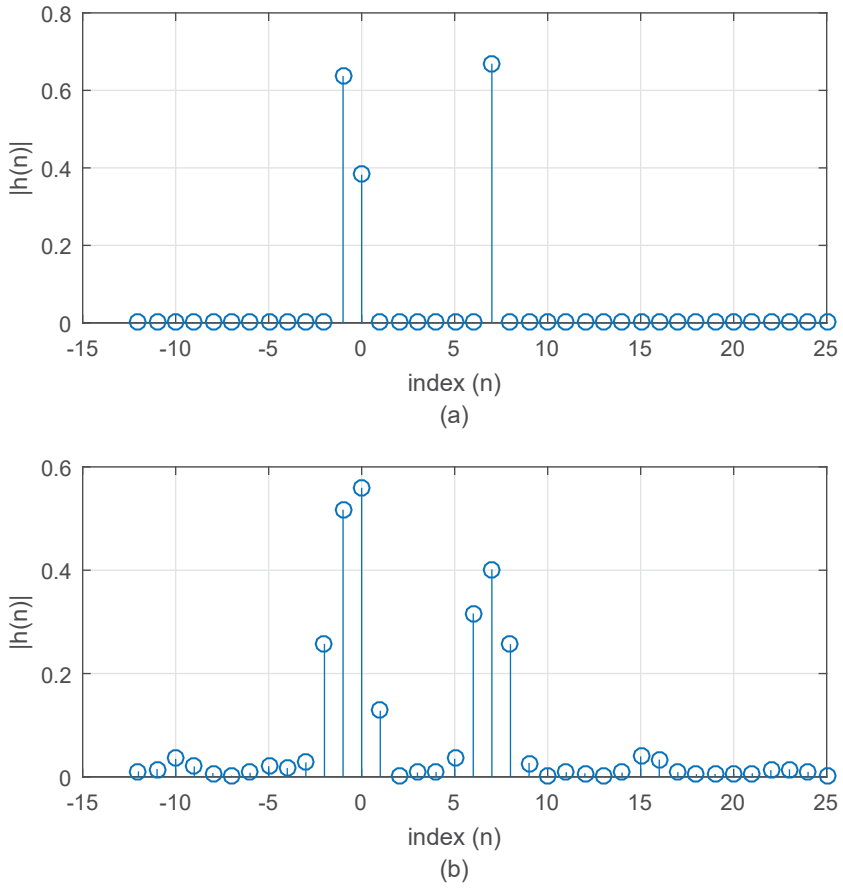


Figure 3.11: True channel measured on Taxiway E; (b) the DS channel estimate at SNR = 16 dB.

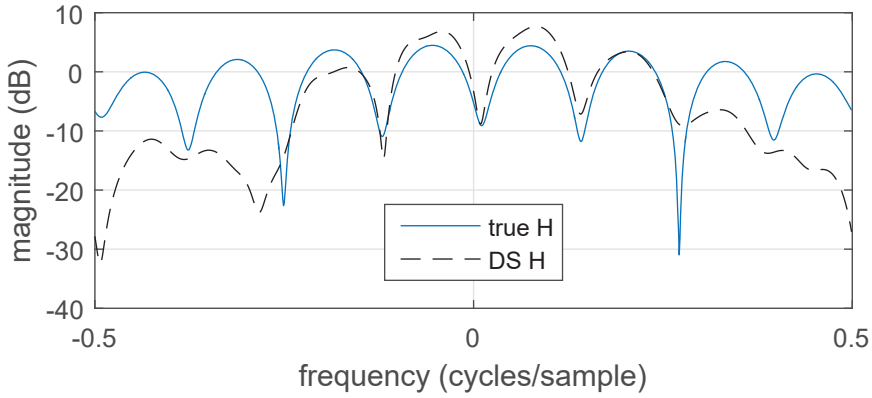


Figure 3.12: The frequency response of the true channel and the DS estimate at SNR = 16 dB.

$$\begin{aligned}
\mathbf{y} &= \mathbf{U}\mathbf{S}\mathbf{V}^H\mathbf{h} + \mathbf{z} \\
\mathbf{U}^H\mathbf{y} &= \mathbf{S}\mathbf{V}^H\mathbf{h} + \mathbf{U}^H\mathbf{z} \\
\mathbf{P}\mathbf{U}^H\mathbf{y} &= \mathbf{P}\mathbf{S}\mathbf{V}^H\mathbf{h} + \mathbf{P}\mathbf{U}^H\mathbf{z} \\
\tilde{\mathbf{y}} &= \mathbf{B}\mathbf{h} + \mathbf{v},
\end{aligned} \tag{3.16}$$

where $\tilde{\mathbf{y}} = \mathbf{P}\mathbf{U}^H\mathbf{y}$, $\mathbf{B} = \mathbf{P}\mathbf{S}\mathbf{V}^H$, and $\mathbf{v} = \mathbf{P}\mathbf{U}^H\mathbf{z}$. An estimate is formed by solving the equation

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{h}\|_1 \quad \text{such that} \quad \mathbf{B}\mathbf{h} = \tilde{\mathbf{y}}. \tag{3.17}$$

The MATLAB implementation is given in Table 3.6. For this application there are 20 non-zero singular values. The time domain estimate and its frequency response are shown in Figures 3.13 and 3.13 respectively.

3.1.4 Greedy Pursuit Algorithms

Greedy pursuit algorithms solve problems iteratively. At each iteration the algorithms select the solution that is locally optimal. This is where the name greedy comes from as the algorithms pick the best solution for the current iteration, but may not select a solution which is globally optimal. Solutions produced by greedy pursuit algorithms approximate a globally optimum solutions.

Greedy pursuit algorithms share a basic structure explained here. The estimate is typically initialized to zero, $\hat{\mathbf{h}} = \mathbf{0}$, the residual vector is initialized to the received samples, $\mathbf{r} = \mathbf{y} - \mathbf{A}\mathbf{h} = \mathbf{y}$, and the support set (the vector of indices corresponding to the nonzero elements in $\mathbf{h}$) is initialized to null, $\mathcal{T} = \emptyset$. The algorithms then begin their cycle of adding elements to $\mathcal{T}$ and updating the estimate $\hat{\mathbf{h}}$, until some stopping criterion is met. This section contains the most popular greedy pursuit algorithms for sparse estimation, but is not an exhaustive list.

Table 3.6: Sparse SVD Estimation

```
function [ h_hat ] = Sparse_SVD_estimate( L, y, A, k )
        k = 20;
        [A_height, ~] = size(A);

        %create selector matrix
        P_diag = [ones(k,1);zeros(A_height-k,1)];
        P = diag(P_diag);

        [U,S,V] = svd(A);
        y_tilda = P*(U'*y);
        beq = [real(y_tilda);imag(y_tilda)];
        B = P*(S*V');

        h0 = zeros(2*L,1);
        L = length(h0);
        for idx = 1:L
                phi(idx*2-1,idx)=1;
                phi(idx*2-1,L+idx) = -1;
                phi(idx*2,idx) = -1;
                phi(idx*2,L+idx) = -1;
        end
        b = zeros(2*L,1);

        B_r = [real(B) -imag(B); imag(B) real(B)];
        phi_eq = [B_r zeros(length(B_r),L)];

        beq = [real(y_tilda);imag(y_tilda)];
        f = [zeros(L,1);ones(L,1)];
        options = optimset('Display','none'); %suppress exit display
        output = linprog(f,phi,b,phi_eq,beq,[],[],[],options);
        h_hat = output(1:38)+1i*output(39:76);
end
```

Matching Pursuit (MP)

One of the most basic greedy algorithms is Matching Pursuit (MP) [20] [7]. Table 3.7 contains the MATLAB function which implements MP for channel estimation. Concerning the three variable that are passed in, L is the length of the channel estimate (Q), $\mathbf{y}$ is the vector $(q \times 1)$ of received samples, and $\mathbf{A}$ is the sensing matrix $(q \times Q)$. The variable $\hat{\mathbf{h}}$ is the channel estimate and is initialized to the zero vector. The variable `denoma` contains the diagonal elements of the matrix produced by $\mathbf{A}^H\mathbf{A}$ and is used in line 8 to compute the squared scalar projection of $\mathbf{y}$ onto

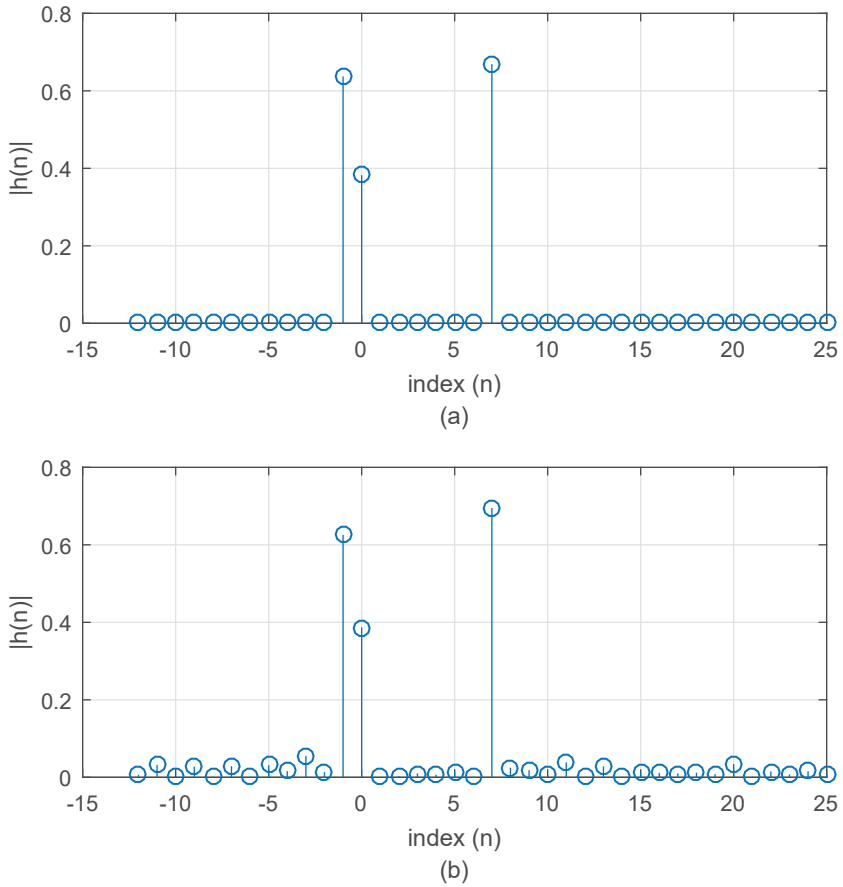


Figure 3.13: True channel measured on Taxiway E; (b) the sparse SVD channel estimate at SNR = 16 dB.

each of the columns of $\mathbf{A}$. The function searches through the squared scalar projections for the largest value. Once the largest value is found, the function calculates the scale factor defining the projection of $\mathbf{y}$ onto the corresponding column of $\mathbf{A}$. The estimate is then updated as shown on line 11. Note, the function updates one coefficient at a time and that the same coefficient can be updated multiple times. The projection of $\mathbf{y}$ onto the selected column of $\mathbf{A}$ is removed from the residual vector, $\mathbf{r}$, and the algorithm repeats until some stopping criterion is met. Here the stopping criterion is six iterations to insure that no more than six filter coefficients are estimated.

Figure 3.15 shows the true channel measured from Taxiway E vs the MP estimate at SNR 16 dB. It is readily apparent that the MP estimate is sparse, something that can not be said for the ML estimate. In this example the MP only estimated five coefficients, instead of six. This means

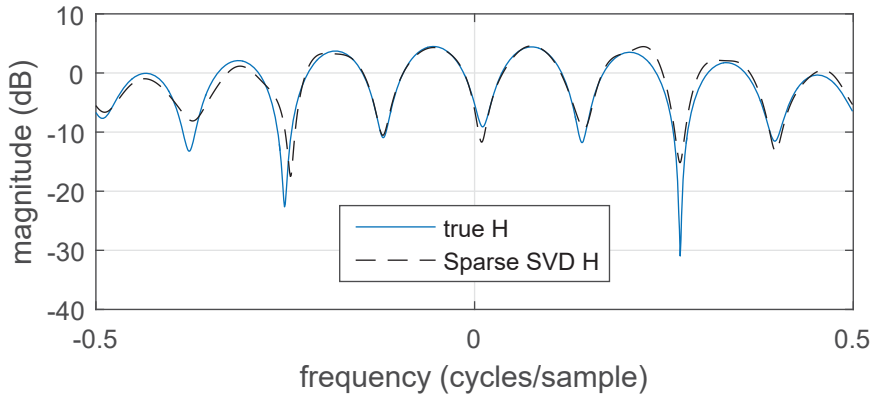


Figure 3.14: The frequency response of the true channel and the sparse SVD estimate at SNR = 16 dB.

Table 3.7: Matching Pursuit Algorithm

```
function h_hat = MP_ChannelEstimate(L ,y, A)
        h_hat = zeros(L,1);
        r = y;
        denoma = diag(A'*A);

        for p = 1:6
                g = A'*r;
                marge = (g.*conj(g))./denoma;
                [~,jj] = max(marge);
                marge =  g(jj)/denoma(jj);
                h_hat(jj) = h_hat(jj) + marge;
                r = r - A(:,jj)*marge;
        end
end
```

the algorithm reselected one of the columns of **A**. Notice, the location of the strongest taps are correctly estimated. However, more important than the time domain estimate of the channel is the frequency response of the estimate. This was shown previously by the ML estimate. Figure 3.16 is the frequency response of the true channel measured from Taxiway E vs the frequency response of the MP estimate at SNR 16 dB. Notice that the frequency response for the estimate is almost perfectly aligned with the frequency response of the true channel between the interval of -0.2 to 0.2 cycles/sample, making it a good estimate.

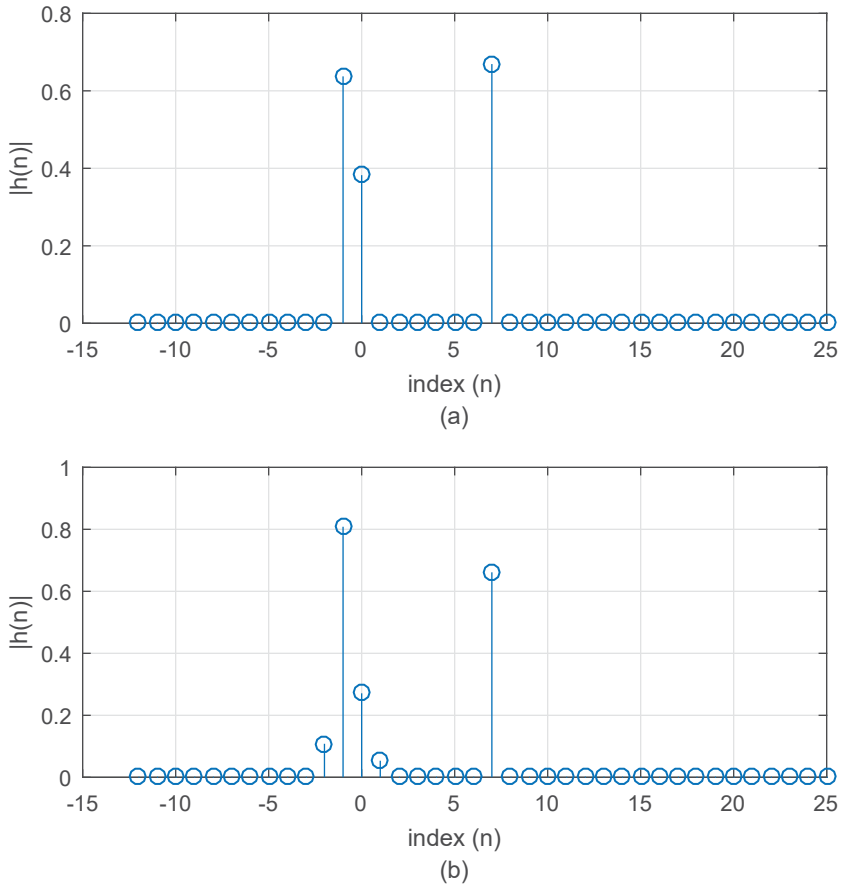


Figure 3.15: True channel measured on Taxiway E; (b) the MP channel estimate at SNR = 16 dB.

Orthogonal Matching Pursuit (OMP)

Table3.8 shows MATLAB code for a variation of the MP algorithm called Orthogonal Matching Pursuit (OMP) [21]. The selection criteria is the same for both the MP and the OMP. The difference is that once a column of $\mathbf{A}$ is selected, it is stored in $\mathcal{T}$ and used in every iteration. The estimate is computed by multiplying the received samples by the left pseudo inverse of the columns of $\mathbf{A}$ defined by $\mathcal{T}$. The residual vector is then updated as shown on line 15. Because the OMP uses the pseudo inverse to compute the estimate $\hat{\mathbf{h}}$, the residual vector is orthogonal to the selected columns of $\mathbf{A}$. This insures that a column of $\mathbf{A}$ will not be selected more than once. Note, unlike MP, OMP estimates the same number of coefficients as selected columns. The greatest dis-

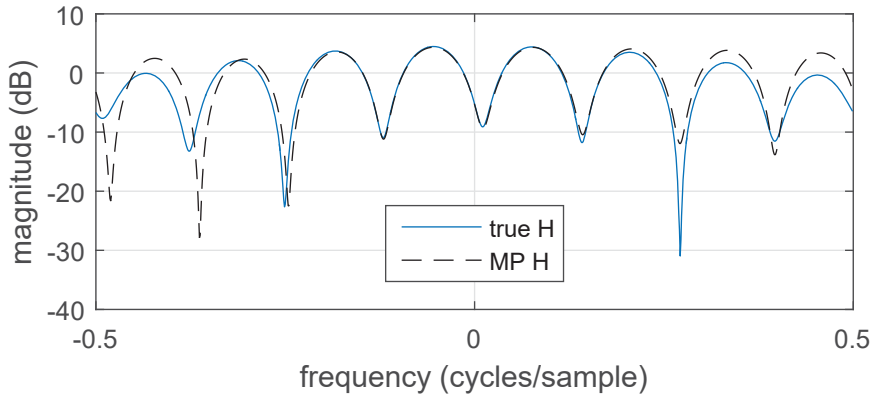


Figure 3.16: The frequency response of the true channel and the MP estimate at SNR = 16 dB.

advantage of using OMP is computational complexity. OMP computes a pseudo inverse for each iteration.

Table 3.8: Orthogonal Matching Pursuit Algorithm

```
function h_hat = CS_ChannelEstimateOMP(L, y, A)

        h_hat = zeros(L,1);
        j = zeros(width(2),1);
        t = zeros(20,1); %vector of indexs
        r = y;
        denoma = diag(A'*A);%width x 1

        for p =1:6 %estimate 6 taps of the filter
                g = A'*r;
                j = (g.*conj(g))./denoma;
                [~, index] = max(j);
                t(p) = index;
                h_hat(t(1:p)) = pinv(A(:,t(1:p)))*y;
                r = y-A*h_hat;
        end
end
```

Figure 3.17 displays the true channel and the OMP estimate at an SNR of 16 dB. Once again, the channel estimate is sparse, only estimating six coefficients. The corresponding frequency re-

sponses of both the true channel and sparse estimate are shown in Figure 3.18. Just like the MP, the OMP channel response is accurate between the interval -0.2 and 0.2 cycles/sample.

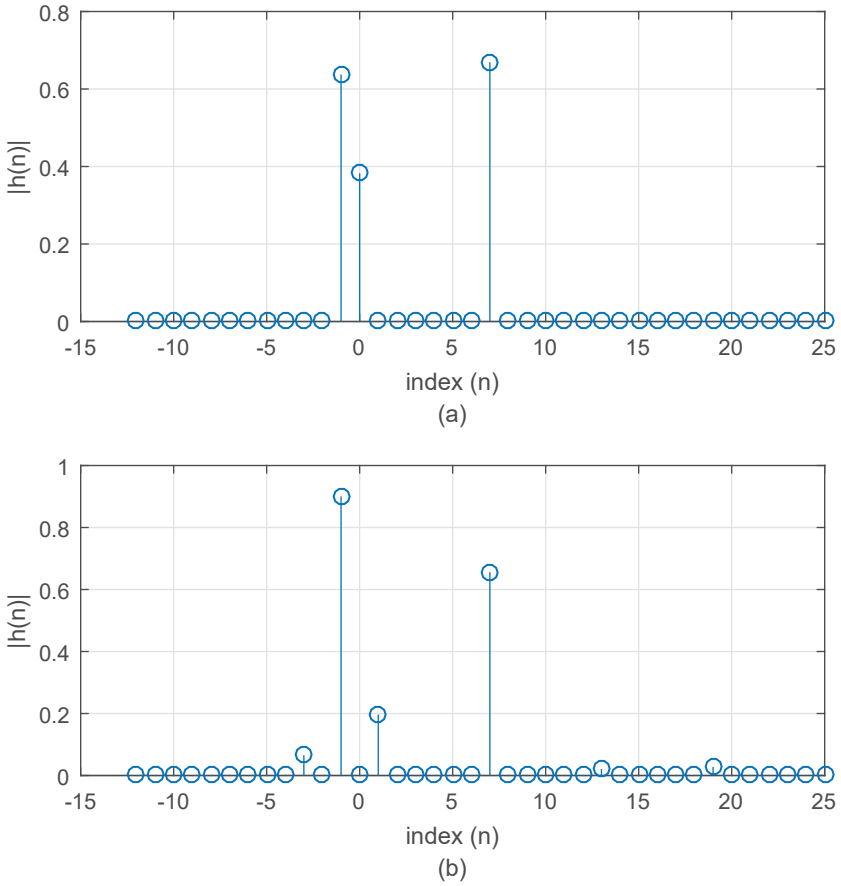


Figure 3.17: True channel measured on Taxiway E; (b) the OMP channel estimate at SNR = 16 dB.

Gradient Pursuit (GP)

The Gradient Pursuit (GP) algorithm is descibed in [19]. MATLAB code implementing GP is shown in Table 3.9.

The GP algorithm is similar to MP, they have same selection criteria and both algorithms can select a column of $\mathbf{A}$ more than once. However, GP stores the selected columns of $\mathbf{A}$ in $\mathcal{T}$ and uses all of the stored columns in each iteration. The GP has been set up as a recursive algorithm where

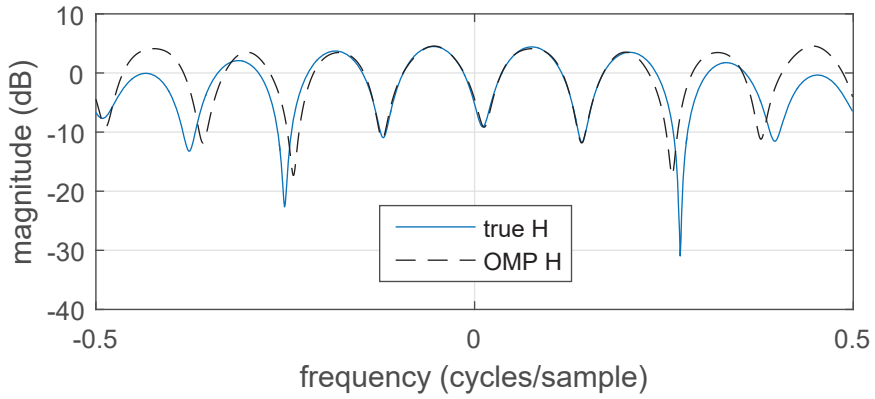


Figure 3.18: The frequency response of the true channel and the OMP estimate at SNR = 16 dB.

the current estimate depends on the previous estimate plus an update as shown on line 18. Here `d` represents the update direction and `a` represents the step size. The update direction is found by taking the negative gradient of the cost function $\|\mathbf{y} - \mathbf{A}_{\mathcal{T}^{[i]}}\mathbf{x}_{\mathcal{T}^{[i]}}\|_2^2$ which is $\mathbf{d}_{\mathcal{T}^{[i]}}^{[i]} = \mathbf{g}_{\mathcal{T}^{[i]}}^{[i]} = \mathbf{A}_{\mathcal{T}^{[i]}}^H(\mathbf{y} - \mathbf{A}_{\mathcal{T}^{[i]}}\hat{\mathbf{h}})$. Luckily we already have this as a result of the selection process and simply need to restrict the vector $\mathbf{g}^{[i]}$ to the elements defined in $\mathcal{T}^{[i]}$. Figure3.19 displays the true channel and the GP estimate at an SNR of 16 dB while Figure 3.20 displays the frequency response of the true channel and the GP channel estimate.

Conjugate Gradient Pursuit (CGP)

An improvement to GP is the Conjugate Gradient Pursuit (CGP) described in [19]. MATLAB code is provided for the CGP implementation in Table 3.10. The CGP uses iterative line minimizations along directions that are G-conjugate to the previous directions where G-conjugate is defined as $< \mathbf{d}^{[i]}, \mathbf{G}\mathbf{d}^{[j]} >= 0$ for $i \neq j$. The direction is initialized to be the gradient, $\mathbf{g}$, shown on line 17 and then calculated as an update as shown on line 21. By carefully selecting `beta`, we can force the direction, $\mathbf{d}^{[i]}$, to be conjugate to all previous directions. Doing so would create an algorithm similar to the QR implementation of OMP. Instead, we have chosen `beta` so that the new direction is only conjugate to the last preceding direction. A computational advantage over OMP is gained

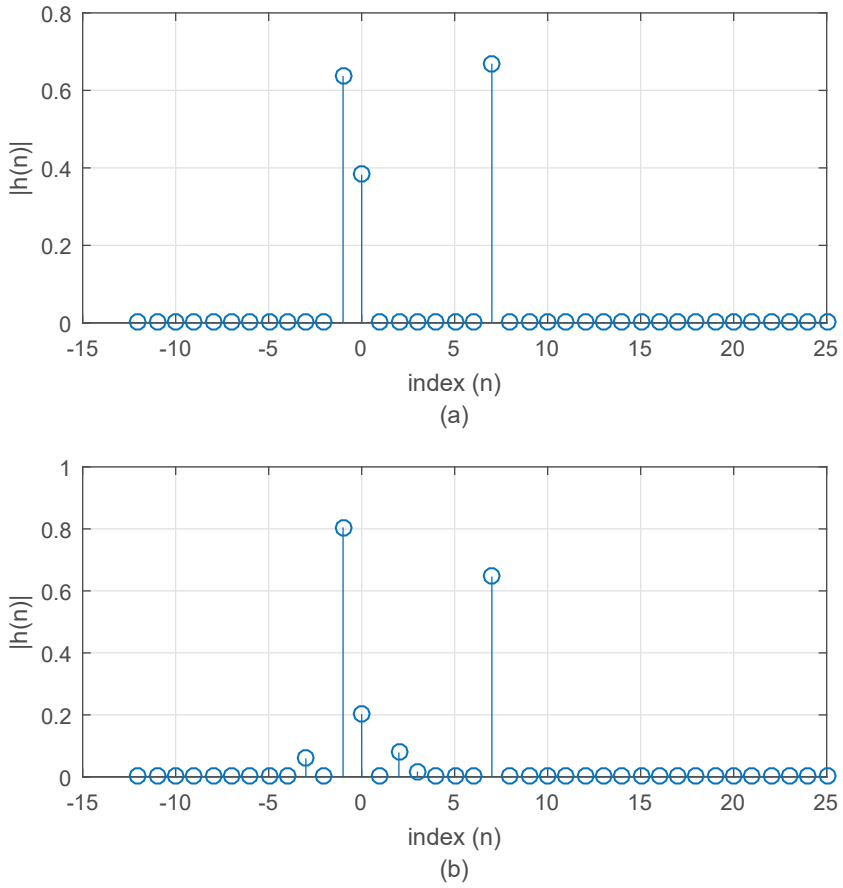


Figure 3.19: True channel measured on Taxiway E; (b) the GP channel estimate at SNR = 16 dB.

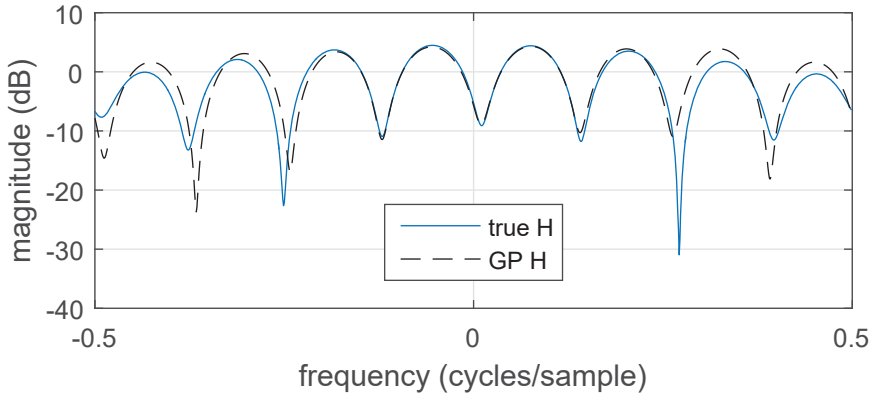


Figure 3.20: The frequency response of the true channel and the GP estimate at SNR = 16 dB.

Table 3.9: Gradient Pursuit Algorithm

```
function h_hat = GDP_CS_ChannelEstimate( L, y , A, k)
        r = y;
        h_hat = zeros(L,1);
        t = 0;
        denoma = diag(A'*A); % width x 1

        for idx = 1:k
                d  = A'*r;
                marge = (g.*conj(g))./denoma;
                [~,jj] = max(marge);
                if idx ==1
                        t = jj;
                end
                t = unique([t,jj]);
                d = g(t);
                c = A(:,t)*d;
                a = c'*r/(c'*c);
                h_hat(t) = h_hat(t) + a*d;
                r = r - a*c;
        end
end
```

because the pseudo inverse is not calculated. Figures 3.21 and 3.22 are the time domain estimates at SNR 16 dB and frequency responses respectively.

Generalized Orthogonal Matching Pursuit (GOMP)

The generalized orthogonal matching pursuit (GOMP) modifies the OMP algorithm [12]. The OMP algorithm selects one column at a time to use in the update of the estimate, the column corresponding to the largest squared scalar projection of the residual vector onto each of the columns of $\mathbf{A}$. GOMP selects k columns to update the estimate in each iteration. The k columns selected correspond to the largest k squared scalar projections of the received vector $\mathbf{y}$ onto each of the columns of $\mathbf{A}$. By letting $k = 6$ and only performing one iteration the GOMP estimates six coefficients and performs one pseudo inverse making it significantly more efficient than OMP. Under these conditions, GOMP turns into an approximation to the brute force method, where the selected columns of $\mathbf{A}$ used to approximate the channel may not be ideal. Table 3.11 is a Matlab implemen-

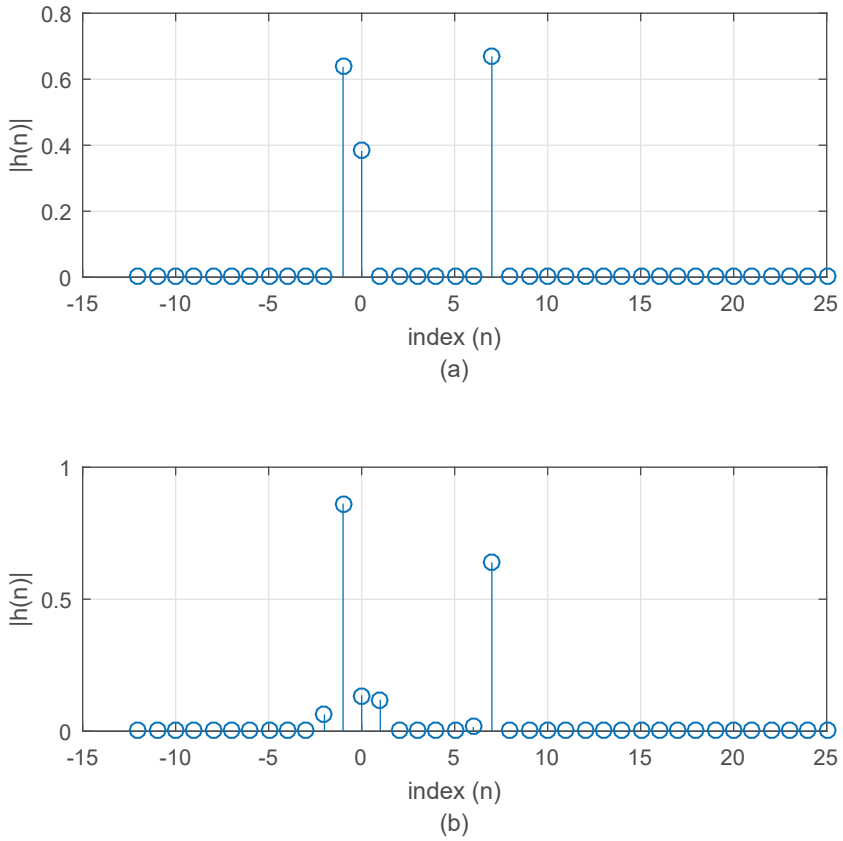


Figure 3.21: True channel measured on Taxiway E; (b) the CGP channel estimate at SNR = 16 dB.

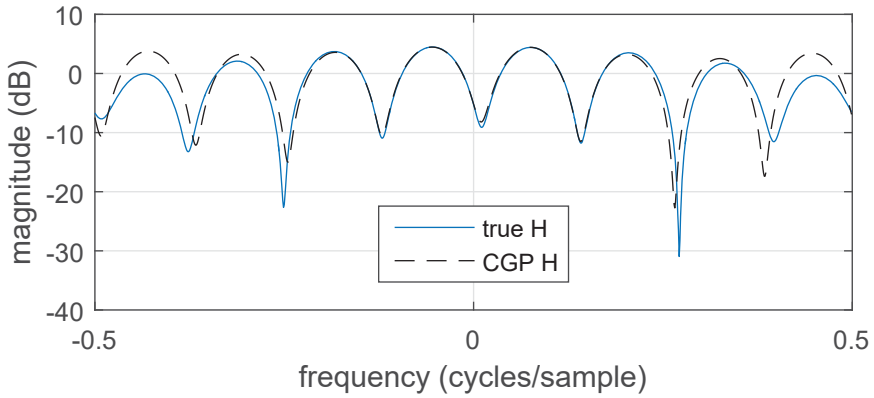


Figure 3.22: The frequency response of the true channel and the CGP estimate at SNR = 16 dB.

Table 3.10: Conjugate Gradient Pursuit Algorithm

```
function h_hat = CGP_CS_ChannelEstimate( L, y , A, k)

        r = y;
        h_hat = zeros(L,1);
        t = 0;
        denoma = diag(A'*A); % width x 1

        for idx = 1:k
                g = A'*r;
                marge = (g.*conj(g))./denoma;
                [~,jj] = max(marge);
                if idx ==1
                        t = jj; %first time through need to initialize it.
                end
                t = unique([t,jj]);
                if idx == 1
                        d = g;
                else
                        c = A(:,t_old)*d(t_old);
                        beta = ((A(:,t)*g(t))'*c)/(c'*c);
                        d(t) = g(t)+beta*d(t);
                end
                c = A(:,t)*d(t);
                a = (c'*r)/(c'*c);
                h_hat(t) = h_hat(t) + a*d(t);
                r = r - a*c;
                t_old = t;
        end
end
```

tation of GOMP. Figure 3.23 displays the true channel and the GOMP estimate. This algorithm closely matches the true channel. The frequency responses of the true channel and GOMP estimate are displayed in Figure 3.24.

3.1.5 Thresholding Algorithms

While greedy pursuit algorithms are fast and easy to implement they do not have recovery guarantees as strong as methods based on convex relaxation. Thresholding algorithms are the in the middle. Typically thresholding algorithms are fast and easy to implement but still have strong recovery guarantees [7]. In this section we will focus on three common thresholding algorithms

Table 3.11: Generalized Orthogonal Matching Pursuit

```
function h_hat = GOMP_CS_ChannelEstimate(L, y, A)
        h_hat = zeros(L,1);
        r = y;
        denoma = diag(A'*A);%width x 1
        g = A'*r;
        j = (g.*conj(g))./denoma;
        [~, index] = sort(j,'descend');
        t = index(1:6); %only estimate 6 taps
        h_hat(t) = pinv(A(:,t))*y;
end
```

used in sparse estimation, the Normalized Iterative Hard Thresholding (NIHT), the Compressive Sampling Matching Pursuit (CoSaMP), and the Subspace Pursuit (SP).

Normalized Iterative Hard Thresholding (NIIT)

NIHT is a greedy algorithm based on the Iterative Hard Thresholding (IHT) algorithm devoloped in [17]. Thus it is useful to start with IHT. IHT solves an approximation to the problem

$$\min_{\mathbf{h}} \|\mathbf{y} - \mathbf{A}\mathbf{h}\|_2^2 \quad \text{subject to} \quad \|\mathbf{h}\|_0 \leq k. \tag{3.18}$$

The surrogate objective function that is minimized by IHT is

$$C_k(x, z) = \mu\|\mathbf{y} - \mathbf{A}\mathbf{h}\|_2^2 - \mu\|\mathbf{A}\mathbf{h} - \mathbf{A}\mathbf{z}\|_2^2 + \|\mathbf{h} - \mathbf{z}\|_2^2. \tag{3.19}$$

Minimizing equation (3.19) subject to $\|\mathbf{h}\|_0 \leq k$ produces

$$\hat{\mathbf{h}} = H_k(\mathbf{z} + \mathbf{z} + \mu\mathbf{A}_{\mathcal{T}}(\mathbf{y} - \mathbf{A}\mathbf{z})), \tag{3.20}$$

where $H_k(a)$ is the non linear operator that keeps the k largest values of a and sets all other elements to zero. By letting $\mathbf{z}$ be equal to $\hat{\mathbf{h}}^{[i]}$, equation (3.20) turns into an iterative algorithm.

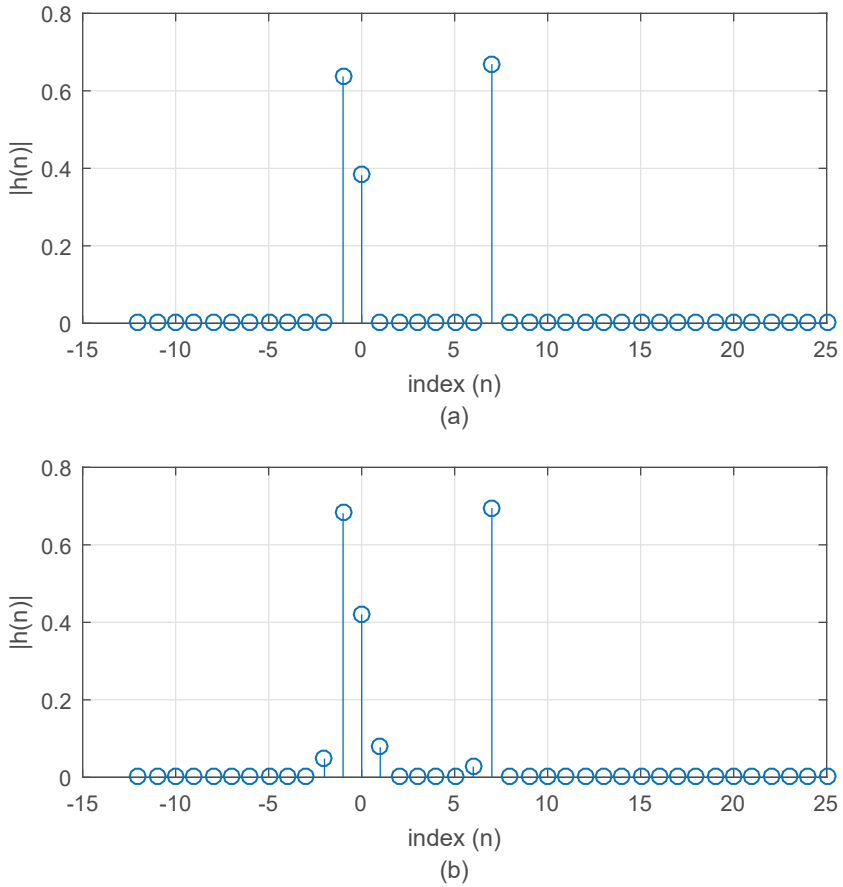


Figure 3.23: True channel measured on Taxiway E; (b) the GOMP channel estimate at SNR = 16 dB.

The IHT is computationally efficient as there are only a few vector multiplications and additions as well as sorting required for thresholding.

In order to guarantee convergence, μ must be selected such that $\beta_{2k} \leq 1/\mu \leq 1.5\alpha_{2k}$, where β_{2k} is defined as the smallest quantity, such that

$$\|\mathbf{A}(\mathbf{h}_1 - \mathbf{h}_2)\|_2^2 \leq \beta_{2k}\|\mathbf{h}_1 - \mathbf{h}_2\|_2^2 \tag{3.21}$$

and α_{2k} is defined in the non-symmetric RIP, which satisfies

$$\alpha_{2k}\|\mathbf{h}_1 - \mathbf{h}_2\|_2^2 \leq \|\mathbf{A}(\mathbf{h}_1 - \mathbf{h}_2)\|_2^2 \leq \beta_{2k}\|\mathbf{h}_1 - \mathbf{h}_2\|_2^2. \tag{3.22}$$

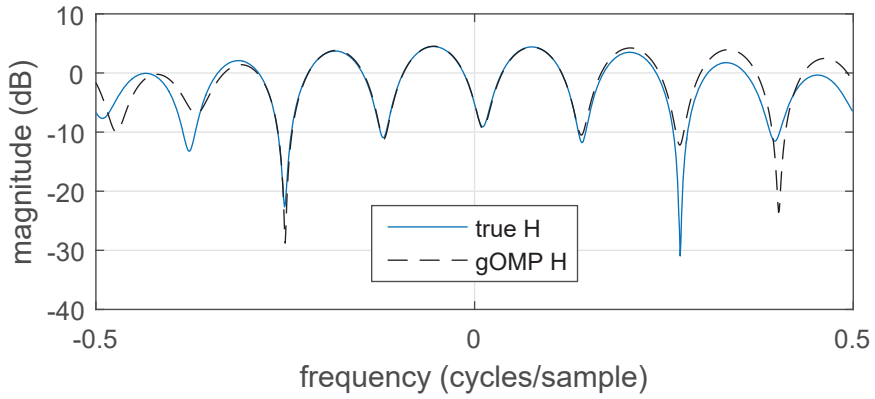


Figure 3.24: The frequency response of the true channel and the GOMP estimate at SNR = 16 dB.

Unfortunately, in most practical cases α_{2k} and β_{2k} are unknown. Furthermore, we would like the algorithm to converge even if $\mathbf{A}$ does not satisfy the RIP conditions.

The solution is to choose μ adaptively as shown in [18]. The result is the NIHT algorithm. Let $\|\mathbf{y} - \mathbf{A}_{\mathcal{T}^{[i]}}\hat{\mathbf{h}}_{\mathcal{T}^{[i]}}\|_2^2$ be the function we want to minimize at each iteration. Then, using a gradient decent algorithm, the estimate update at iteration $i+1$ is $\hat{\mathbf{h}}_{\mathcal{T}^{[i]}}^{[i+1]} = \hat{\mathbf{h}}_{\mathcal{T}^{[i]}}^{[i]} - \mu\mathbf{A}_{\mathcal{T}^{[i]}}^{H}(\mathbf{y} - \mathbf{A}_{\mathbf{t}^{[i]}}\mathbf{h}_{\mathcal{T}^{[i]}}^{[i]})$. From [7] the optimal step size, μ, is

$$\mu = \frac{\|\mathbf{g}_{\mathcal{T}^{[i]}}^{[i]}\|_2^2}{\|\mathbf{A}_{\mathcal{T}^{[i]}}\mathbf{g}_{\mathcal{T}^{[i]}}^{[i]}\|_2^2}, \tag{3.23}$$

where $\mathbf{g}_{\mathcal{T}^{[i]}}^{[i]}$ is the sub-vector of $\mathbf{g}^{[i]}$ defined by $\mathcal{T}^{[i]}$ and $\mathbf{A}_{\mathcal{T}^{[i]}}$ is the sub-matrix of $\mathbf{A}$. The benefit of using (3.23) is that if $\mathbf{A}$ satisfies the non-symmetric RIP conditions then

$$\alpha_{2k} \leq \frac{\|\mathbf{A}_{\mathcal{T}^{[i]}}\mathbf{g}_{\mathcal{T}^{[i]}}^{[i]}\|_2^2}{\|\mathbf{g}_{\mathcal{T}^{[i]}}^{[i]}\|_2^2} \leq \beta_{2k} \tag{3.24}$$

and the function will converge.

If $\mathbf{A}$ does not meet the non-symmetric conditions, can it still be guaranteed that the function will converge? The answer is yes, but now we need a line search approach. The estimate update is set to a temporary variable, i.e. $\tilde{\mathbf{h}} = H_k(\hat{\mathbf{h}}^{[i]} + \mu^{[i]}\mathbf{g}^{[i]})$, and the support of $\tilde{\mathbf{h}}$ is compared to the

support of the previous iteration. If the support remains unchanged, then the estimate $\hat{\mathbf{h}}$ is set to $\tilde{\mathbf{h}}$. If the support changes, then the μ that was computed may no longer be optimal and must be checked to see if it satisfies [18]

$$\mu \leq (1-c)\frac{\|\tilde{\mathbf{h}}^{[i+1]} - \hat{\mathbf{h}}^{[i]}\|_2^2}{\|\mathbf{A}(\tilde{\mathbf{h}}^{[i+1]} - \hat{\mathbf{h}}^{[i]})\|_2^2}, \tag{3.25}$$

where c is some small positive number. If the computed μ satisfies Equation (3.26), then the estimate $\hat{\mathbf{h}}^{[i]}$ is set to the temporary variable $\tilde{\mathbf{h}}^{[i]}$. When μ fails the comparison in (3.25), the algorithm enters into a loop where a new μ is computed by

$$\mu = \frac{\mu}{\kappa(1-c)}, \tag{3.26}$$

for some constant $\kappa > 1/(1-c)$. The temporary variable $\tilde{\mathbf{h}}$ is recomputed using the new μ, and μ is compared to Equation (3.25). The loop is repeated until Equation (3.25) is satisfied. Upon exiting the loop a new support $\mathcal{T}$ is computed and the estimate $\hat{\mathbf{h}}$ is set to the temporary variable $\tilde{\mathbf{h}}$. Table 3.12 contains a MATLAB implementation of NIHT.

Figure 3.25 shows the true channel and the NIHT estimate. While the NIHT algorithm is able to find the location of the strongest taps it is also more susceptible to noise. Some of the true channel coefficients that are zero are estimated to be relatively large by NIHT. As a result, the frequency response of the estimate is not as accurate as shown in Figure 3.26. It is plainly visible that the NIHT estimate is poor at higher frequencies. However, as stated before, the SOQPSK-TG PSD contains most of its energy in the interval -0.25 to 0.25 cycles/sample which might allow NIHT to perform better than the ML estimate.

Compressive Sampling Matching Pursuit (CoSaMP)

The CoSaMP algorithm was first introduced by Needell and Tropp in [16]. At the start of the algorithm, the support is initialized by setting $\mathcal{T} = supp(H_k(\mathbf{A}^H\mathbf{y}))$, where $supp(\cdot)$ returns the support of $(\cdot)$ and the estimate to $\hat{\mathbf{h}} = \mathbf{0}$. In each iteration the residual error $\mathbf{y} - \mathbf{A}\hat{\mathbf{h}}$ is

Table 3.12: Normalized Iterative Hard Thresholding

```
function h_hat = NIHT_CS_ChannelEstimate(L, y, A, k)
        c = .3;
        alpha = 1/(1-c)+10;
        h_hat = zeros(L,1);
        %find the largest k elements and save their indeces
        [z I] = sort(A'*y,'descend');
        t = I(1:k);
        for count = 1:k
                g = A'*(y-A*h_hat);
                mu = norm(g(t),2)^2/norm(A(:,t)*g(t),2)^2;
                [htemp t_temp]= guess(h_hat,mu,g,k);

                %%%check to make sure the new support is the same as the old
                    support
                if(isequal(sort(t,'descend'),sort(t_temp,'descend')))
                        h_hat = htemp;
                else
                        if mu <= compute_mu_thresh(htemp,h_hat,A,c)
                                h_hat = htemp;
                        else
                                while mu < compute_mu_thresh(htemp,h_hat,A,c)
                                        mu = mu/(alpha*(1-c))
                                        [htemp t_temp]= guess(h_hat,mu,g,k);
                                end
                                t = t_temp;
                                h_hat = htemp;
                        end
                end
        end
end

function htemp t_temp = guess(h_hat,mu,g,k)
        htemp = (h_hat+mu*g);
        %find preserve largest k values and set others to zero
        [z I] = sort(htemp,'descend');
        htemp(I(k+1:end))=0;
        t_temp = I(1:k);
end

function mu_thresh = compute_mu_thresh(htemp, h_hat, A, c)
        mu_thresh = (1-c)*norm(htemp-h_hat,2)^2/norm(A*(htemp-h_hat),2)^2;
end
```

projected onto each of the columns of $\mathbf{A}$ to form $\mathbf{g}$. The indexes of the largest $2k$ elements of $\mathbf{g}$ are selected and added to the support $\mathcal{T}^{[i]}$ to create the intermediate support $\mathcal{T}^{[i+0.5]}$. An intermediate

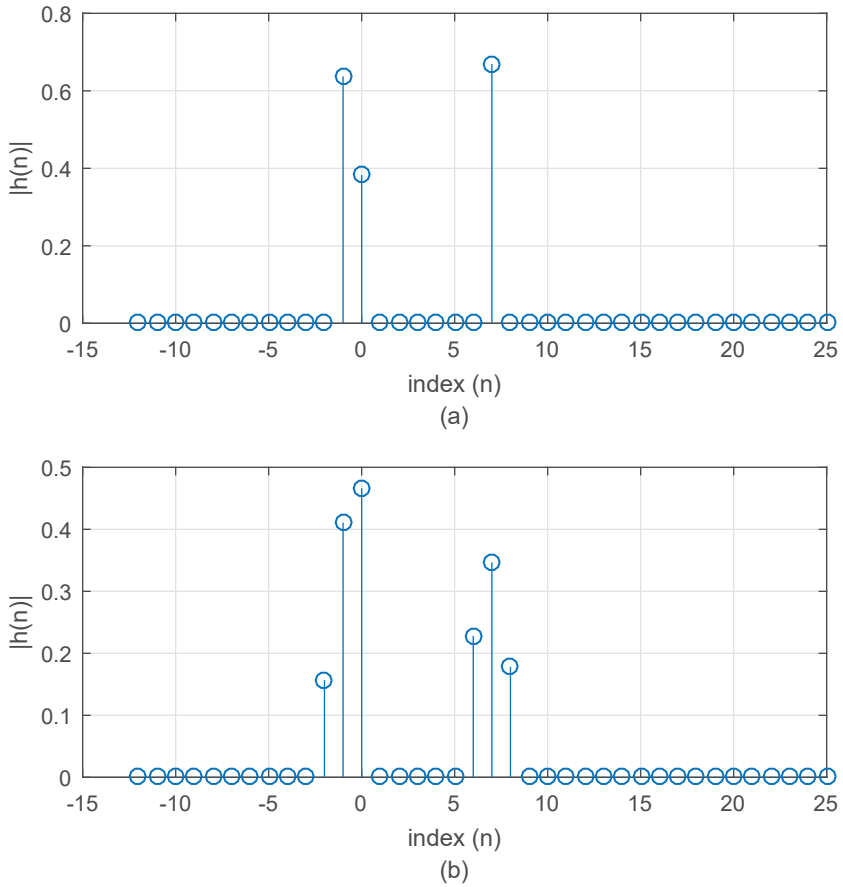


Figure 3.25: True channel measured on Taxiway E; (b) the NIHT channel estimate at SNR = 16 dB.

estimate is formed by $\hat{\mathbf{h}}_{\mathcal{T}^{[i+0.5]}}^{[i+0.5]} = (\mathbf{A}_{\mathcal{T}}^H \mathbf{A}_{\mathcal{T}})^{-1} \mathbf{A}_{\mathcal{T}}^H \mathbf{y}$ and $\hat{\mathbf{h}}_{\overline{\mathcal{T}^{[i+0.5]}}}^{[i+0.5]} = \mathbf{0}$, where $\overline{a}$ represents the compliment of a. The new support is computed by storing the indexes corresponding to the largest k elements of the intermediate estimate i.e. $\mathcal{T}^{[i+1]} = supp(\hat{\mathbf{h}}_k^{[i+0.5]})$. The new estimate is computed by keeping the elements of the intermediate estimate corresponding to the indexes found in $\mathcal{T}^{[i+1]}$, i.e. $\hat{\mathbf{h}}_{\mathcal{T}^{[i+1]}}^{[i+1]} = \hat{\mathbf{h}}_{\mathcal{T}^{[i+1]}}^{[i+0.5]}$. Table 3.13 is the MATLAB implementation of CoSaMP.

The channel estimate for CoSaMP is displayed in Figure 3.27. This algorithm does a poor job estimating the taps correctly and is highly susceptible to noise. The frequency response of the estimate is shown in Figure 3.28. Unlike the previous algorithms that estimated the low frequency terms accurately, the CoSaMP algorithm does not. Because the estimate does not produce an

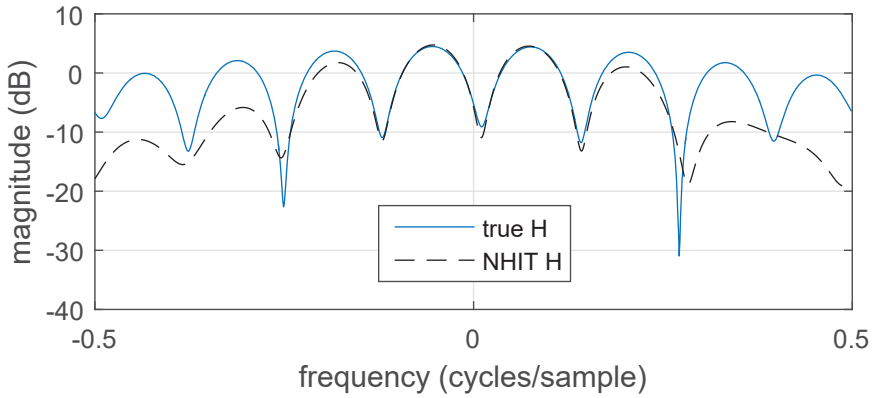


Figure 3.26: The frequency response of the true channel and the NIHT estimate at SNR = 16 dB.

accurate representation of the true channel response, we can hypothesize that this algorithm will perform poorly when compared to the others.

Subspace Pursuit (SP)

The Subspace Pursuit (SP), introduced by Dai and Milenkovic in [13], is similar to the CoSaMP algorithm with a few notable differences. First, is the initialization of the estimate. Most sparse estimation algorithms initialize the estimate to the zero vector. SP, on the other hand, finds the indexes corresponding to the largest k values, of $\mathbf{A}^H\mathbf{y}$ and stores them in $\mathcal{T}^{[0]}$. It then initializes the estimate to $\hat{\mathbf{h}} = (\mathbf{A}_{\mathcal{T}}^H\mathbf{A}_{\mathcal{T}})^{-1}\mathbf{A}_{\mathcal{T}}^H\mathbf{y}$. Second, the intermediate support is the union of the indexes corresponding to the k largest values of $\mathbf{g}$ instead of the $2k$ largest values where $\mathbf{g}$ is the projection of the residual vector $\mathbf{r}$ onto each of the columns of $\mathbf{A}$. Finally, once the new support $\mathcal{T}^{[i+1]}$ is computed a second least-squares estimate is found using $\mathcal{T}^{[i+1]}$ i.e. $\hat{\mathbf{h}}^{[i+1]} = \mathbf{A}_{\mathcal{T}^{[i+1]}}^{\dagger}\mathbf{y}$. Table 3.14 contains Matlab code for SP implementation

Figures 3.29 and 3.30 are the channel estimate and frequency response of the channel estimate for SP respectively. While the SP algorithm is very similar to CoSaMP, it performs much better. The increase in performance can be attributed to the second least squares estimate performed at the end of each iteration in the SP algorithm.

Table 3.13: Compressive Sampling Matching Pursuit

```
function h_hat = CoSaMP_CS_ChannelEstimate(L, y , A, k)
        r = y;
        t = supp(A'*r,k);
        h_hat = zeros(L,1);
        indices = 1:L;
        for idx = 1:k
                g = A'*(r-A*h_hat);
                t_half = unique([t; supp(g,2*k)]); % t union g
                h_hat_half = zeros(L,1);
                h_hat_half(t_half) = pinv(A(:,t_half))*r;
                [~,IA] = setdiff(indices,t_half);
                h_hat_half(IA) = 0;
                t = supp(h_hat_half,k);
                h_hat(t) = h_hat_half(t);
                [~,IA] = setdiff(indices,t);
                h_hat(IA) = 0;
        end
end

        %calculate the support of x with largest k values
function t = supp(x,k)
        [z, I] = sort(x,'descend');
        t = I(1:k);
end
```

3.1.6 Summary

This chapter presented the algorithms used for sparse channel estimation. There are three categories of algorithms; optimization algorithms, greedy pursuits, and thresholding algorithms. Optimization algorithms seek to impose a sparse estimate by either minimizing the L_2 norm using the L_1 norm as a constraint or by minimizing the L_1 norm with a constraint on some defined error. Greedy pursuits are iterative algorithms and use selected columns of $\mathbf{A}$ and the received vector $\mathbf{y}$ to form a sparse estimate. Thresholding algorithms are similar to greedy pursuits, but estimate more than the desired number of non-zero coefficients and then prune, until only the desired number of non-zero coefficients are left.

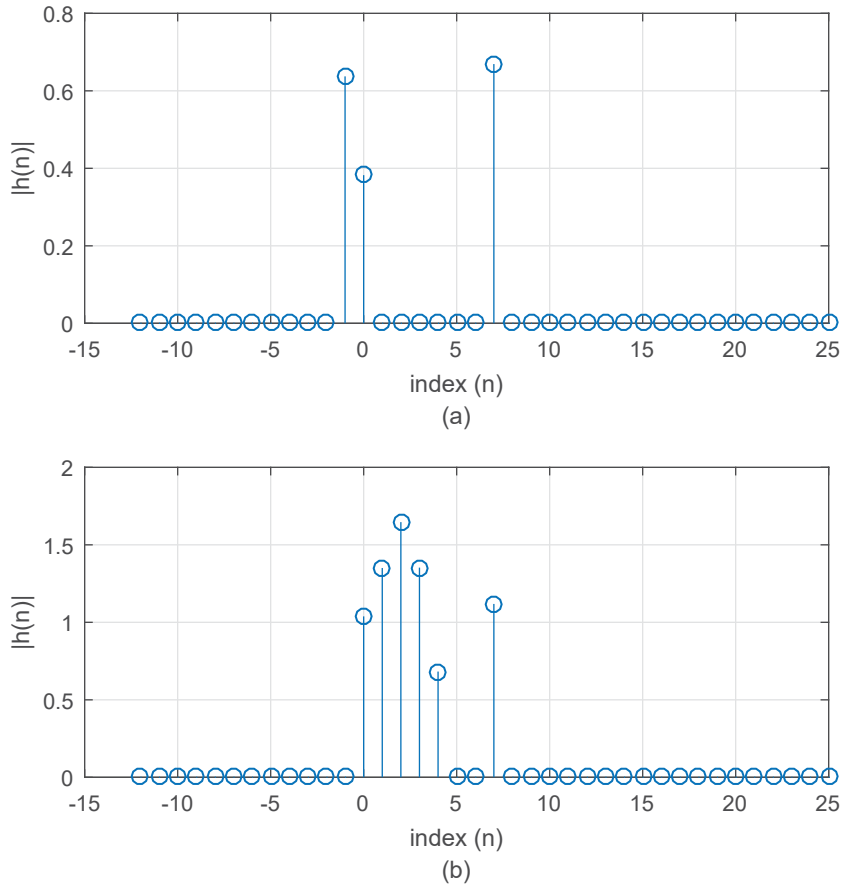


Figure 3.27: True channel measured on Taxiway E; (b) the CoSaMP channel estimate at SNR = 16 dB.

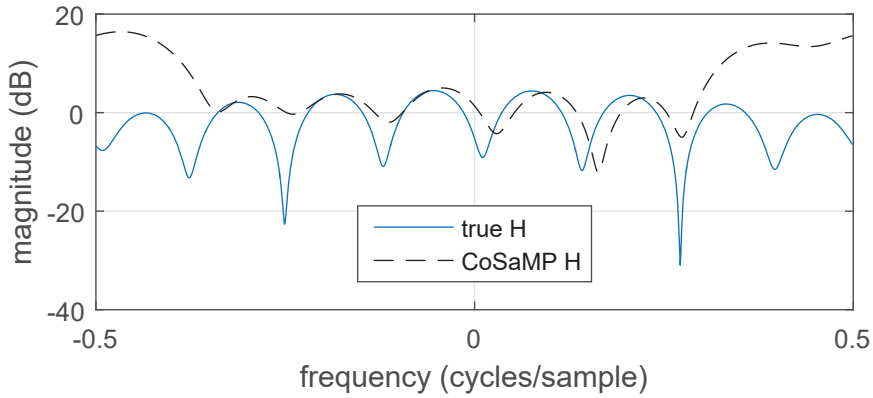


Figure 3.28: The frequency response of the true channel and the CoSaMP estimate at SNR = 16 dB.

Table 3.14: Subspace Pursuit

```
function h_hat = SP_CS_ChannelEstimate( L, y, A,k )
        t = supp(A'*r,k);
        h_hat = zeros(L,1);
        h_hat(t) = pinv(A(:,t)) * y;
        indices = 1:L;
        for idx = 1:k
                g = A'*r;
                t_half = unique([t; supp(g,k)]); % t union g
                h_hat_half = zeros(L,1);
                h_hat_half(t_half) = pinv(A(:,t_half))*y;
                [~,IA] = setdiff(indices,t_half);
                h_hat_half(IA) = 0; % set all elements of h_hat_half at
                % indices not in t_half to zero
                t = supp(h_hat_half,k);
                h_hat(t) = pinv(A(:,t))*y;
                r = y-A*h_hat;
        end
end

        %calculate the support of x with largest k values
function t = supp(x,k)
        [z I] = sort(x,'descend');
        t = I(1:k);
end
```

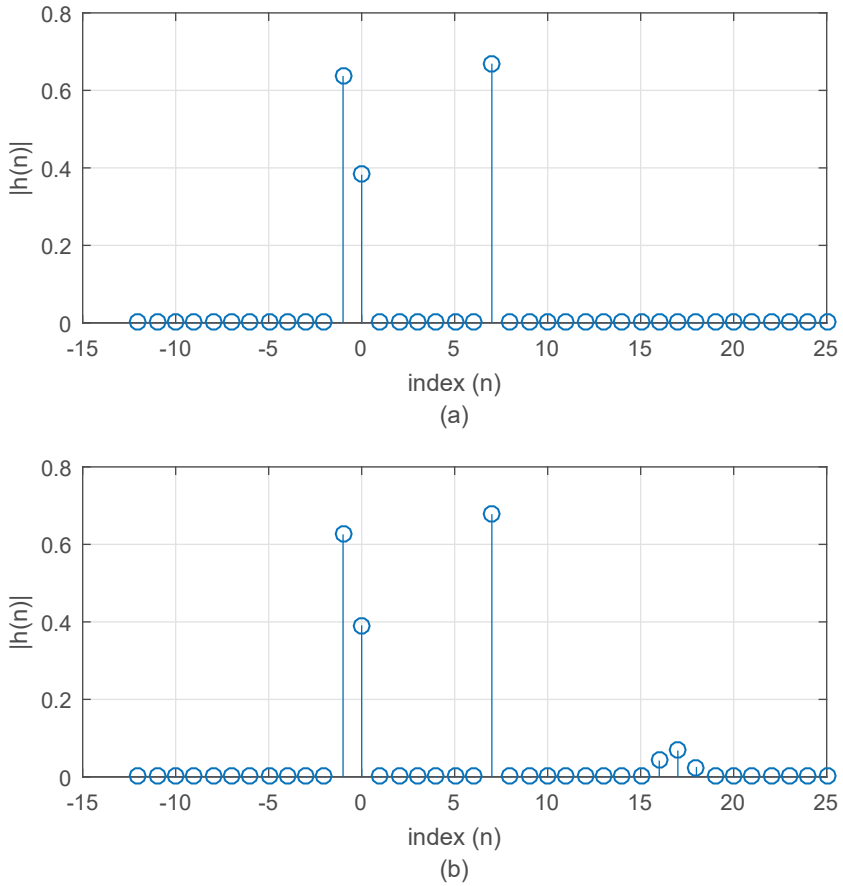


Figure 3.29: (a) True channel measured on Taxiway E; (b) the SP channel estimate at SNR = 16 dB.

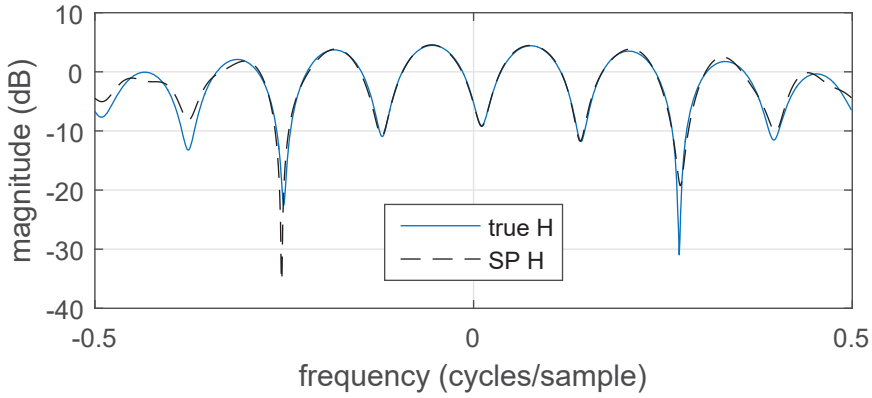


Figure 3.30: The frequency response of the true channel and the SP estimate at SNR = 16 dB.

CHAPTER 4. RESULTS

4.1 BER Simulation Results for no Frequency Offset

Each algorithm outlined in Chapter 3 was tested by implementing the block diagram in Figure 2.3 in MATLAB. An SOQPSK-TG modulator produced a packet of samples, denoted $s(nT)$, at a sample rate of 2 samples/bit. The samples were passed through a multipath channel modeled as an LTI system with impulse response $h(nT)$ and noise was added to form the received samples $r(nT)$. The noise, $z(nT)$ was modeled as a complex-valued circularly symmetric normal random process with zero mean and autocorrelation

$$E\left[z((n+m)T)z^*(nT)\right] = \frac{2N_0}{T}\delta(m). \tag{4.1}$$

An estimate of the channel seen by the equalizer was formed using the waveform samples corresponding to the pilot sequence and one of the algorithms described in Chapter 3. For all of the greedy pursuit and thresholding algorithms we let $k = 6$, meaning a maximum of 6 non-zero coefficients were estimated.

The coefficients of the equalizer filter were found by solving the Weiner-Hopf equations derived from the minimum mean-squared error (MMSE) criterion [25]. The signal estimate $\hat{s}(nT)$ was formed by applying the MMSE equalizer to the received samples. An SOQPSK-TG demodulator converted $\hat{s}(nT)$ into bit decisions and a comparator computed the bit errors between the orignal data and the demodulated data.

The performance of the algorithms was measured by the BER at the demodulator output as a function of SNR. The BERs for each sparse estimation algorithm are presented for each channel in the next four sections. All of the channels were sampled at 2 samples/bit [6].

4.1.1 Channel 1 (Taxiway E, Edwards AFB)

Channel 1 was measured at Edwards Airforce Base on Taxiway E. Figure 4.1 is the impulse response of the channel and Figure 4.2 is the frequency response and group delay of the channel. There are three non-zero coefficients in Channel 1, two being adjacent and one separated by six zeros. Channel 1 suffers from many nulls in the frequency domain, as well as severe group delay distortion.

The BERs for Channel 1 are shown in Figure 4.3. The curve labeled GENIE is the post-equalizer BER, when the equalizer uses the true channel instead of a channel estimate. The brute force method is not listed, because of the computational time required to form an estimate. Though it is difficult to see from the plots, the GOMP, SPARSEVA-RE and LASSO algorithms produce the curves closest to the GENIE curve. All of algorithms except for CoSaMP, DP, DS, and the sparse SVD estimation algorithm, outperform the ML estimate by 0-1 dB.

The CoSaMP algorithm performs poorly for all tested SNRs. This is explained as follows: CoSaMP begins by estimating the locations and amplitudes of the $3k$ largest coefficients, using the pseudo-inverse of $3k$ columns of $\mathbf{A}$. This initial estimate is pruned to reduce the number and location of channel coefficients (see lines 13 and 14 of Table 3.13). Because the initial estimate uses a large number of columns of $\mathbf{A}$, the initial estimate is similar to the ML estimate. The example shown in Section 1.3 demonstrated that the largest coefficients in the ML estimate do not necessarily correspond to the best sparse estimate (see Figure 1.4). The end result is a channel estimate that may not be a good sparse estimate of the true channel.

The sparse SVD estimation algorithm occasionally produces a bad channel estimate. When this occurs the BER is nearly 0.5 for a full payload of data. As the SNR increases, the BER is increasingly dominated by these bad channel estimates and the curve flattens out.

The DS erroneously produces large non-zero estimates at locations where the true coefficients are zero. These incorrect coefficients adversely affect the frequency response of the estimate. Thus the BER for DS is dominated by the poor estimates.

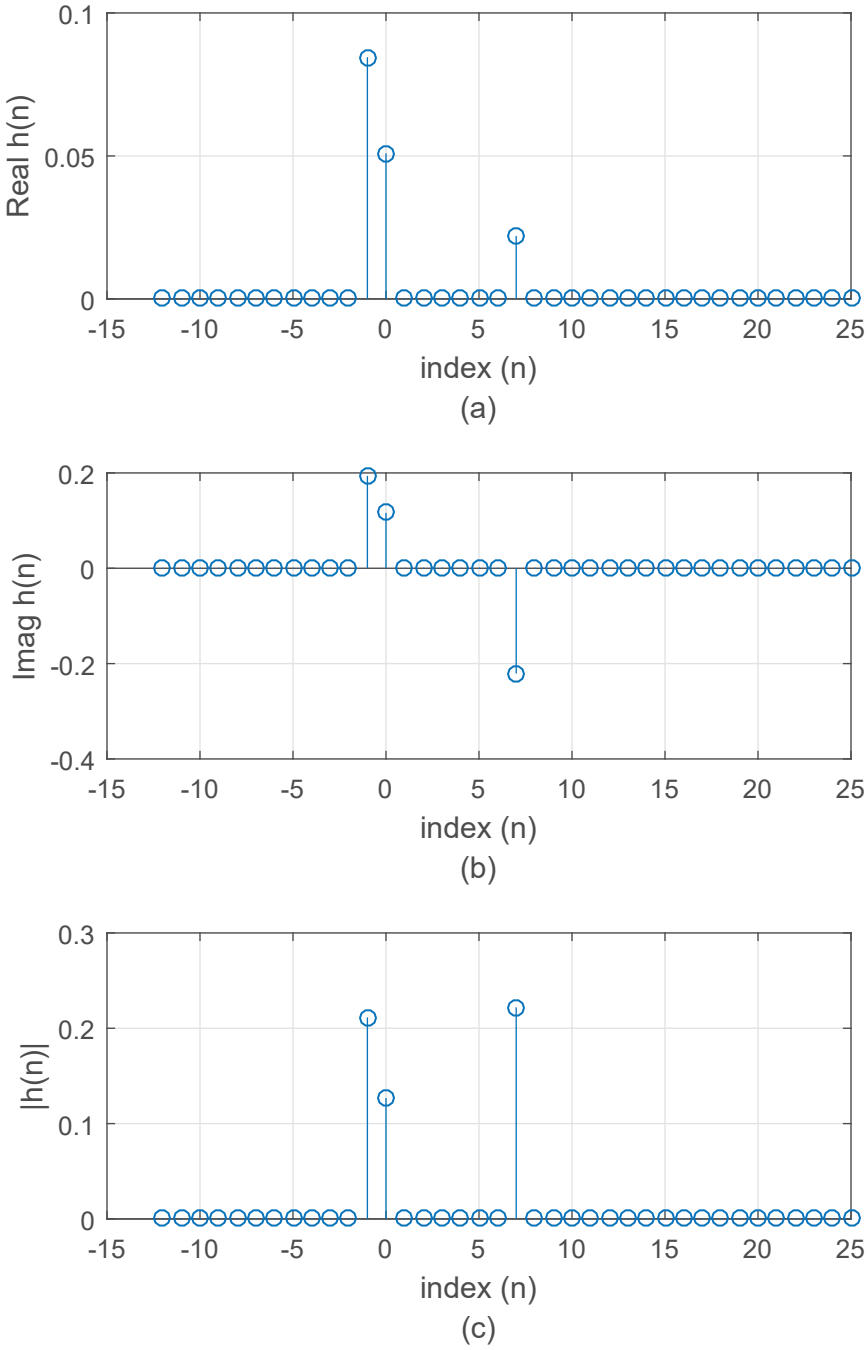


Figure 4.1: The impulse response of the channel measured on Taxiway E at Edwards AFB; (a) is a graph of the real coefficients; (b) is a graph of the imaginary coefficents; (c) is a graph of the magnitude of the coefficients.

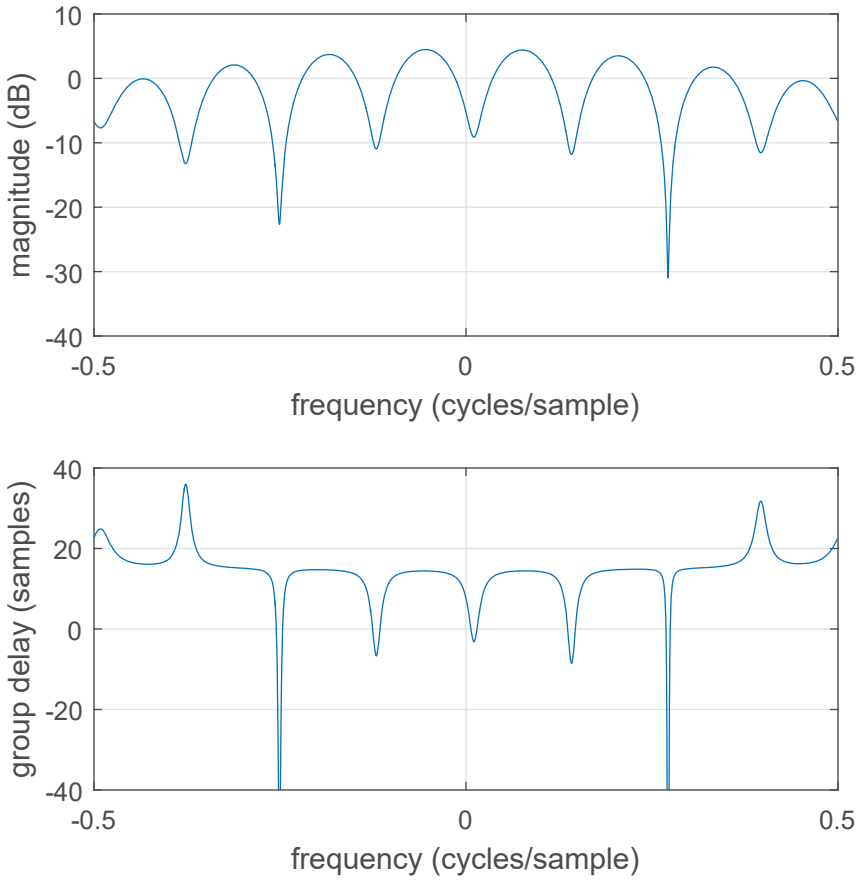


Figure 4.2: (a) is the magnitude of the frequency response in dB; (b) is the group delay of the channel.

The DP algorithm performs poorly at high SNRs, because the tuning parameter is not chosen dynamically, with SNR. In [9], Dai and Pelkmans formulate the problem with unit-variance noise and examine estimator performance, as a function of the number of rows of the sensing matrix. The system under consideration keeps the number of rows of the sensing mtrix constant and varies SNR. At low SNR, the tuning parameter should be large. In contrast, the tuning parameter should be small at high SNR. It is not clear how the tuning parameter should be adjusted with SNR.

4.1.2 Channel 2 (Black Mountain Flight Corridor, Edwards AFB)

Channel 2 was measured on the Black Mountain flight corridor at Edwards AFB. The channel impulse response is shown in Figure 4.4 and the frequency response and group delay are shown

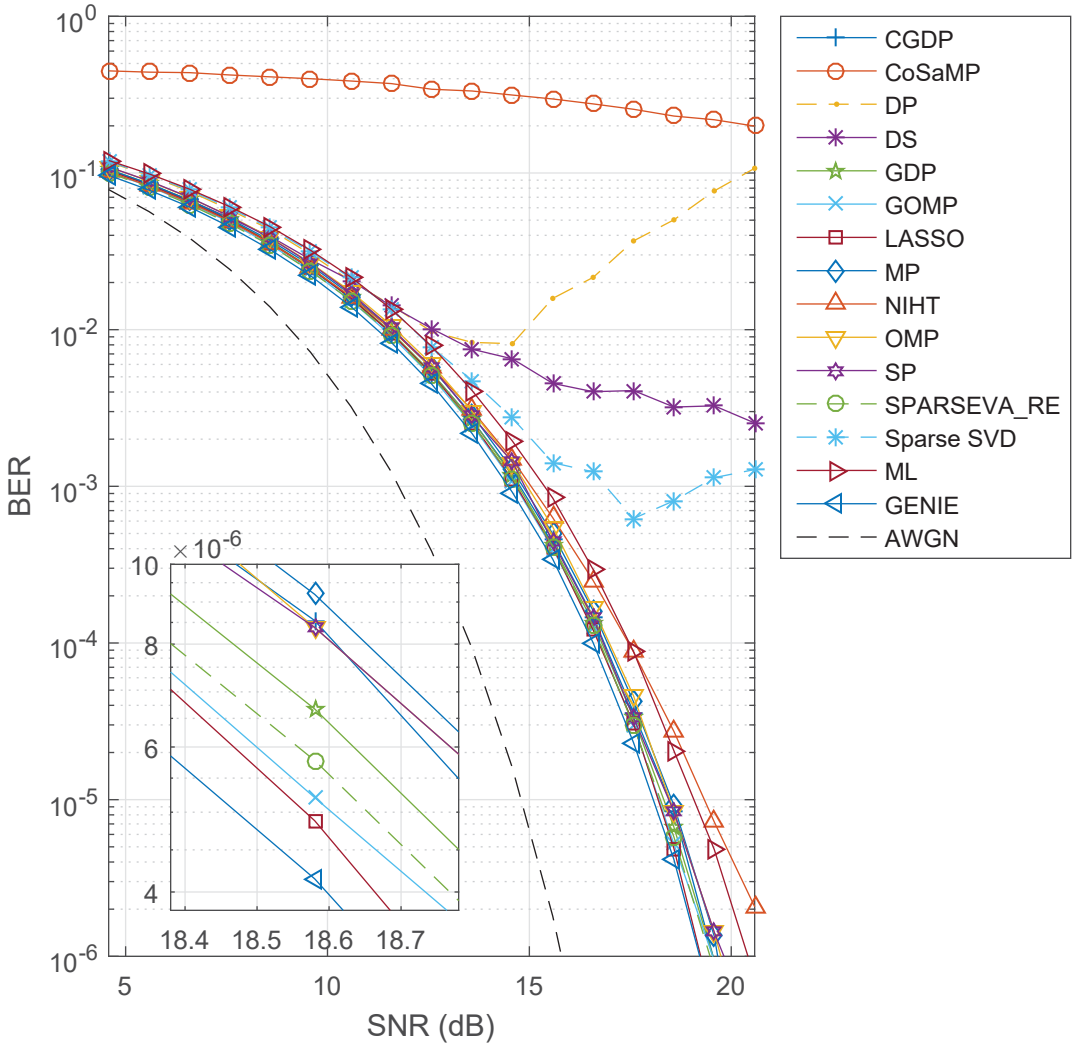


Figure 4.3: The BER curves for each of the algorithms when equalizing the channel measured on Taxiway E at Edwards AFB (Channel 1).

in Figure 4.5. Unlike Channel 1, the location of the non-zero coefficients of Channel 2 are close together with one strong coefficient and two smaller coefficients. There is only one shallow null in the frequency response and the group delay is flat.

The BER curves for channel 2 are shown in Figure 4.6. The BER curves for this channel are similar to Channel 1, with the exception that DS now outperforms the ML estimate.

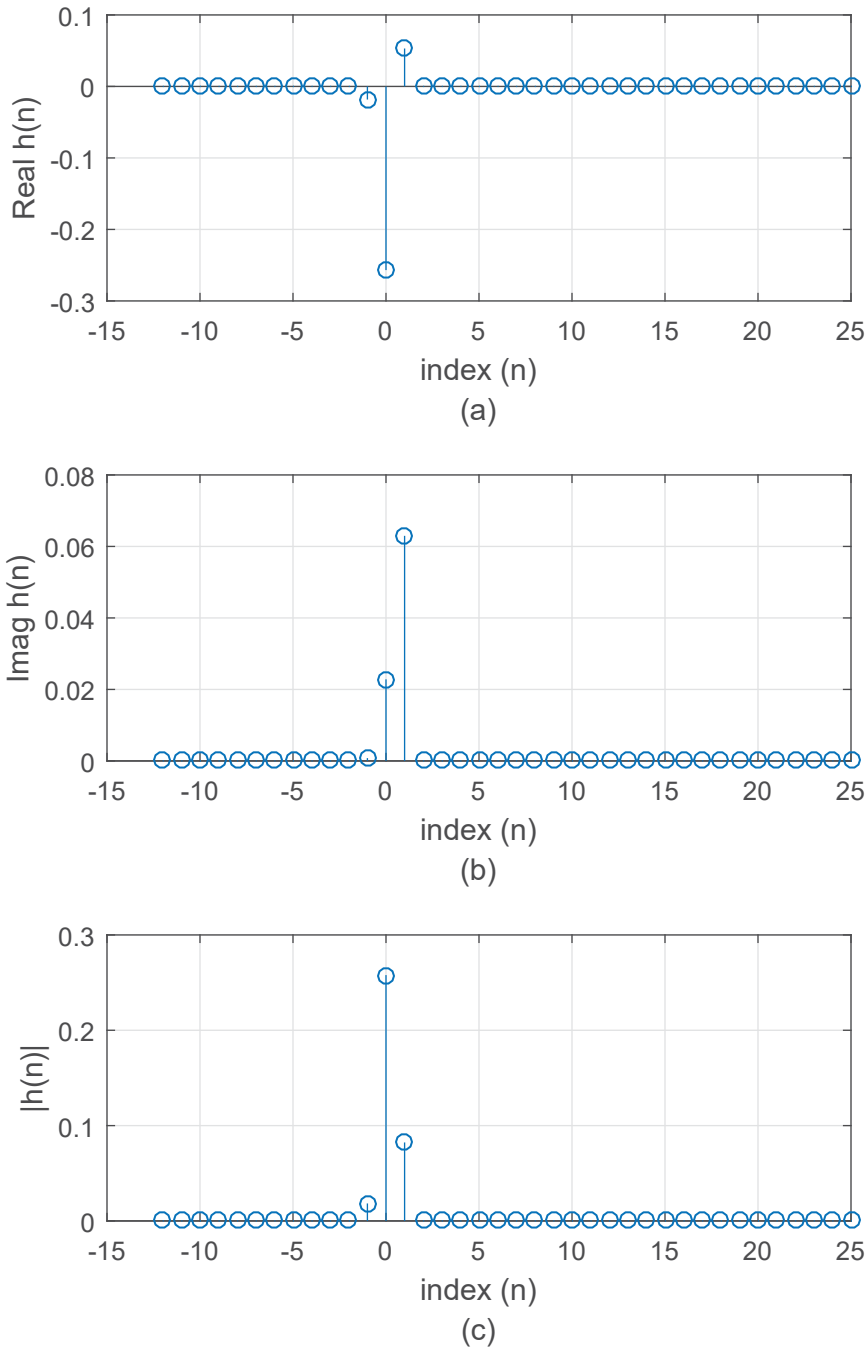


Figure 4.4: The impulse response of the channel measured on the Black Mountain flight corridor at Edwards AFB; (a) is a graph of the real coefficients; (b) is a graph of the imaginary coefficients; (c) is a graph of the magnitude of the coefficients.

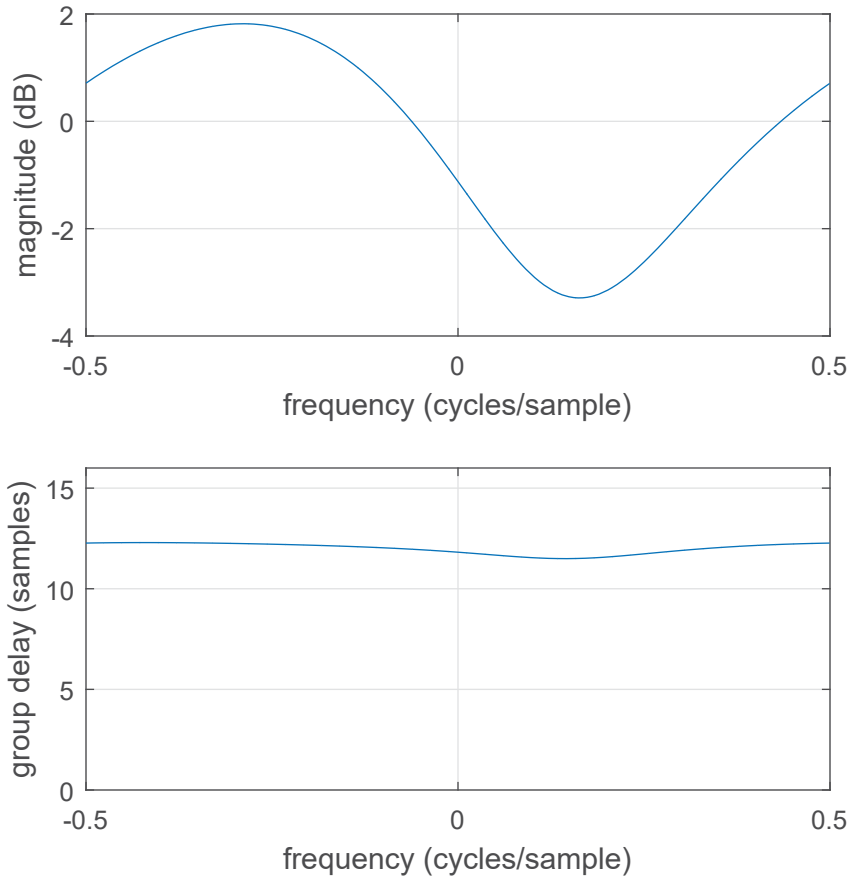


Figure 4.5: (a) is the magnitude of the frequency response of Channel 2 in dB; (b) is the group delay of the channel.

4.1.3 Channel 3 (Cords Road Flight Corridor, Edwards AFB)

Channel 3 was measured on Cords Road flight corridor at Edwards AFB. The impulse response of the channel is shown in Figure 4.10. The magnitude of the impulse response shows that there are two strong coefficients and 3 smaller ones. The frequency response and group delay of Channel 4 appear to be similar to the frequency response and group delay of Channel 3.

The BER for Channel 3, shown in Figure 4.9, follows the same pattern as the BERs for Channel 2.

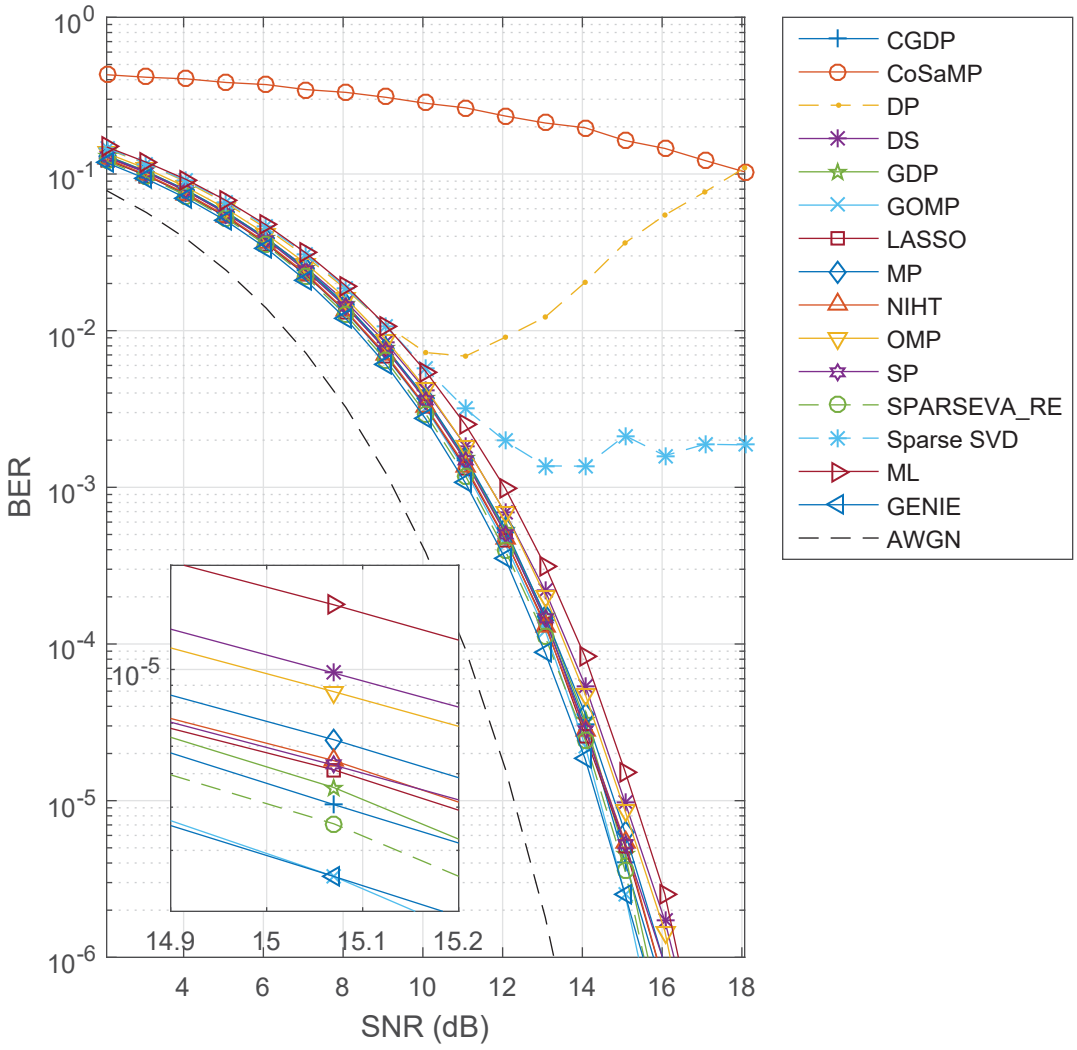


Figure 4.6: The BER curves for each of the algorithms when equalizing the channel measured on the Black Mountain flight corridor at Edwards AFB (Channel 2).

4.1.4 Channel 4 (Final Approach/Landing on Runway 22L, Edwards AFB)

Channel 4 was measured on Final Approach/Landing on 22L. The channel impulse response is shown in Figure 4.7. The frequency response and group delay are shown in Figure 4.8. There are six non-zero coefficients located close together. The frequency response and group delay of the channel each exhibit one null.

The BER curves for Channel 4 are shown in Figure 4.12. All of the algorithms except for NIHT performed the same as on Channel 1. The NIHT performs poorly under high SNR for this channel.

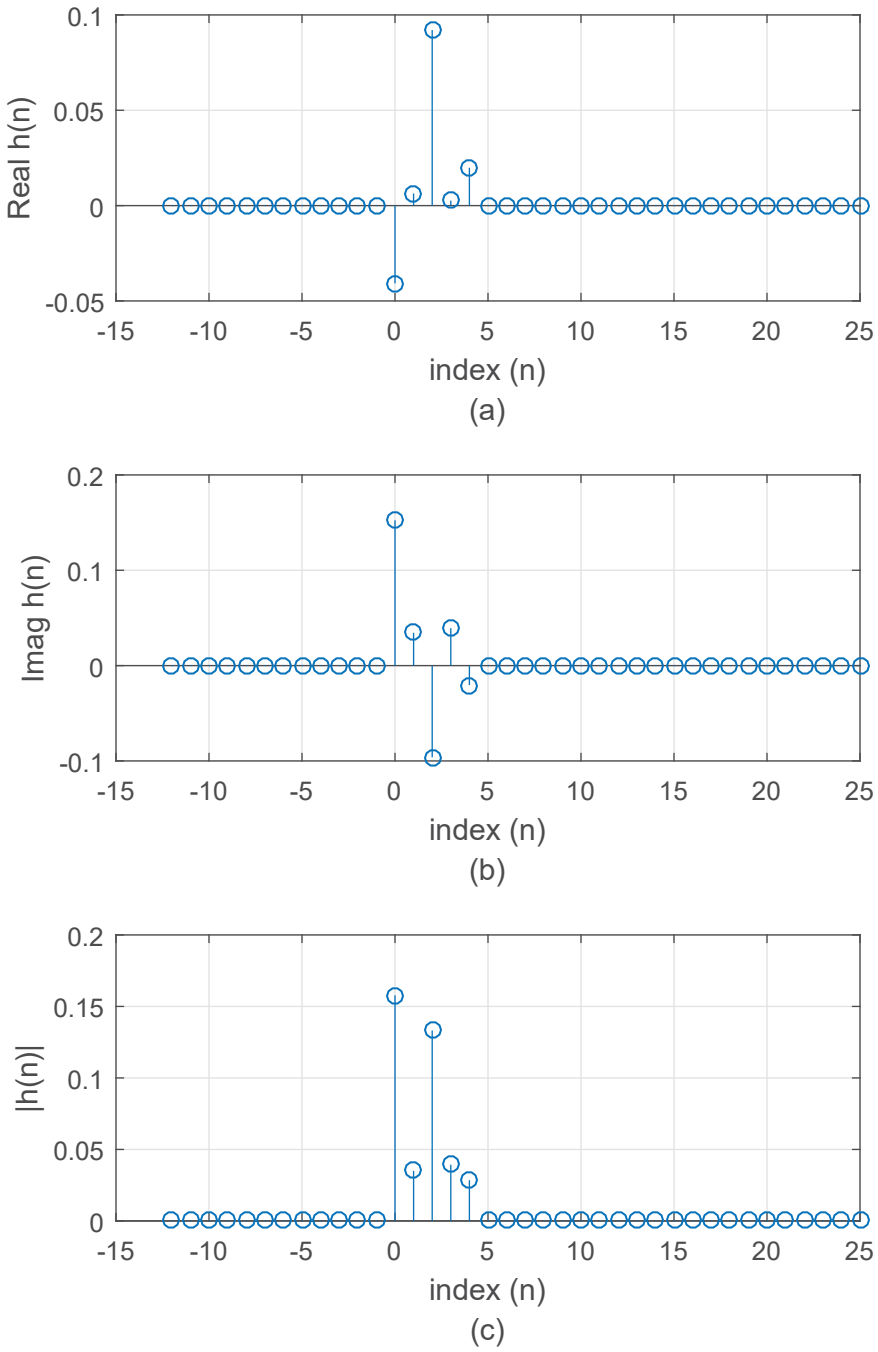


Figure 4.7: The impulse response of the channel measured on the Cords Road flight corridor at Edwards AFB; (a) is a graph of the real coefficients; (b) is a graph of the imaginary coefficients; (c) is a graph of the magnitude of the coefficients.

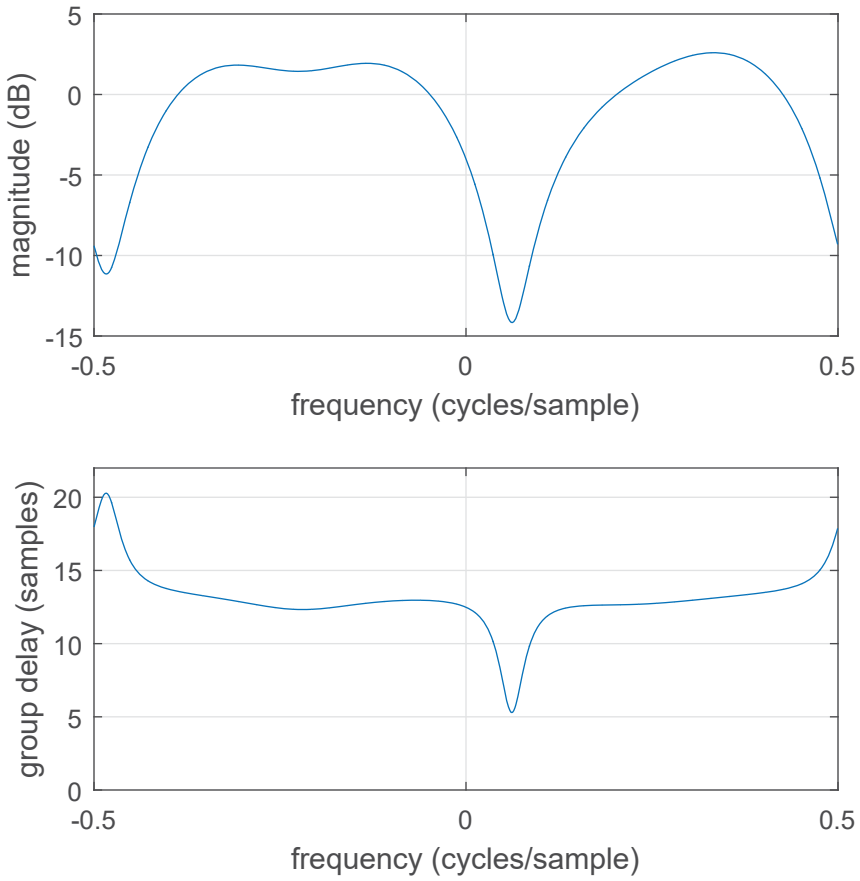


Figure 4.8: (a) is the magnitude of the frequency response of channel 3 in dB; (b) is the group delay of the channel.

There are two factors contributing to its lack of performance under high SNRs. First is the number of coefficients the NIHT algorithm estimates. As stated earlier, the number of coefficients to be estimated is k, regardless of whether the channel has less coefficients. The second issue is how the estimate is formed. NIHT makes an initial estimate using the gradient vector $\mathbf{g}$, which is the projection of the vector of received samples $\mathbf{y}$ onto each of the columns of the sensing matrix $\mathbf{A}$. To form the estimate, NIHT scales $\mathbf{g}$ by the step size μ and keeps the largest k values, setting the rest to zero. If the true channel has only k' non-zero coefficients, then the largest k' values of $\mathbf{g}$ correspond to the non-zero coefficients. However, when $k' < k$ there are $k - k'$ non-zero coefficients that are kept and, while they do not equal the k' true non-zero coefficients in magnitude, they can be quite

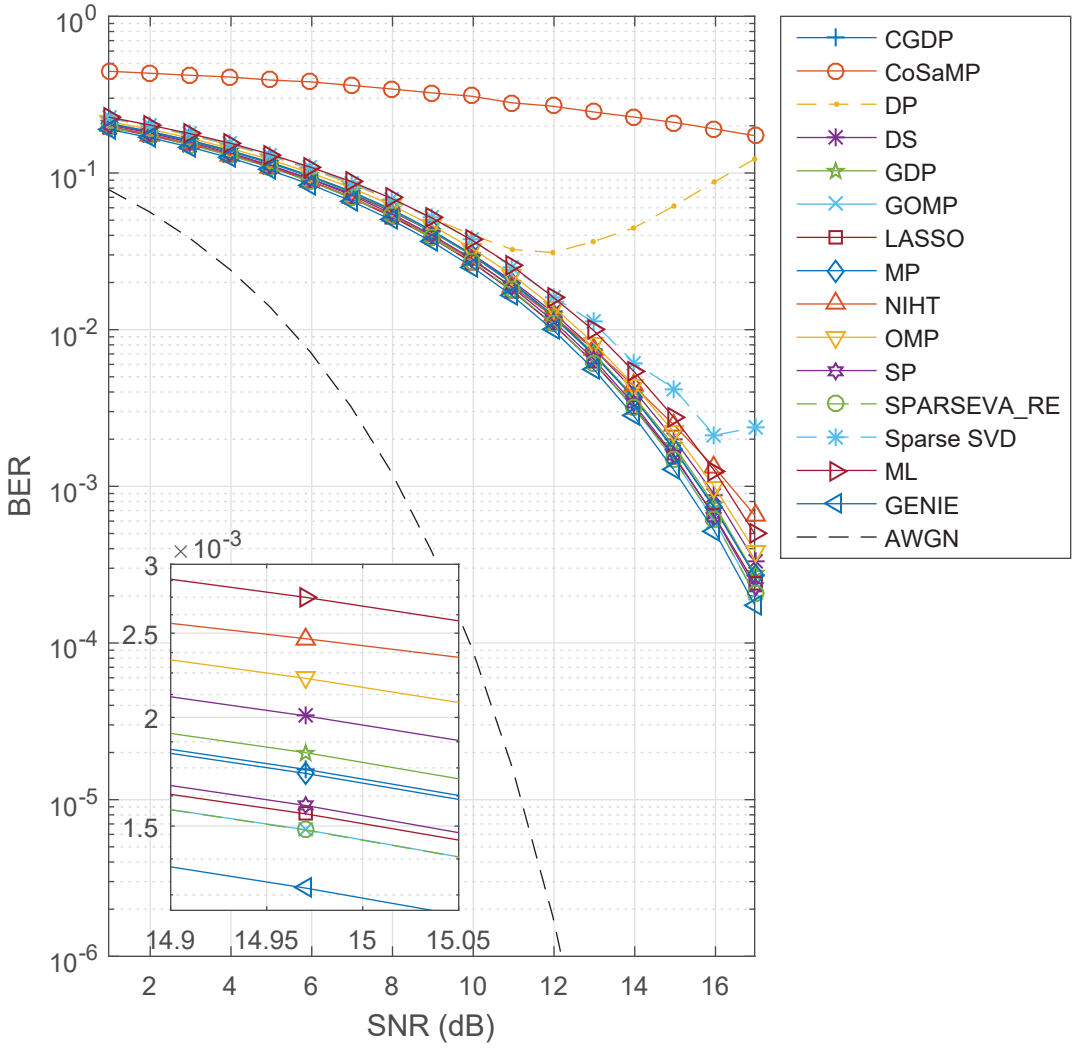


Figure 4.9: The BER curves for each of the algorithms when equalizing the channel measured on the Cords Road flight corridor at Edwards AFB (Channel 3).

large. Estimating the exact number of non-zero coefficients greatly enhances the performance of the NIHT algorithm, but this is not always known a priori.

4.2 BER Simulation Results with Frequency Estimation Errors

The previous section assumed no frequency offset between the transmitter and receiver. However, in most systems there is a small frequency offset which must be corrected with a first order phase lock loop (PLL). Figure 4.13 is a block diagram of the symbol by symbol (SxS) detector and PLL used in this system. We refer the reader to [26] for a description of the SxS detector. The

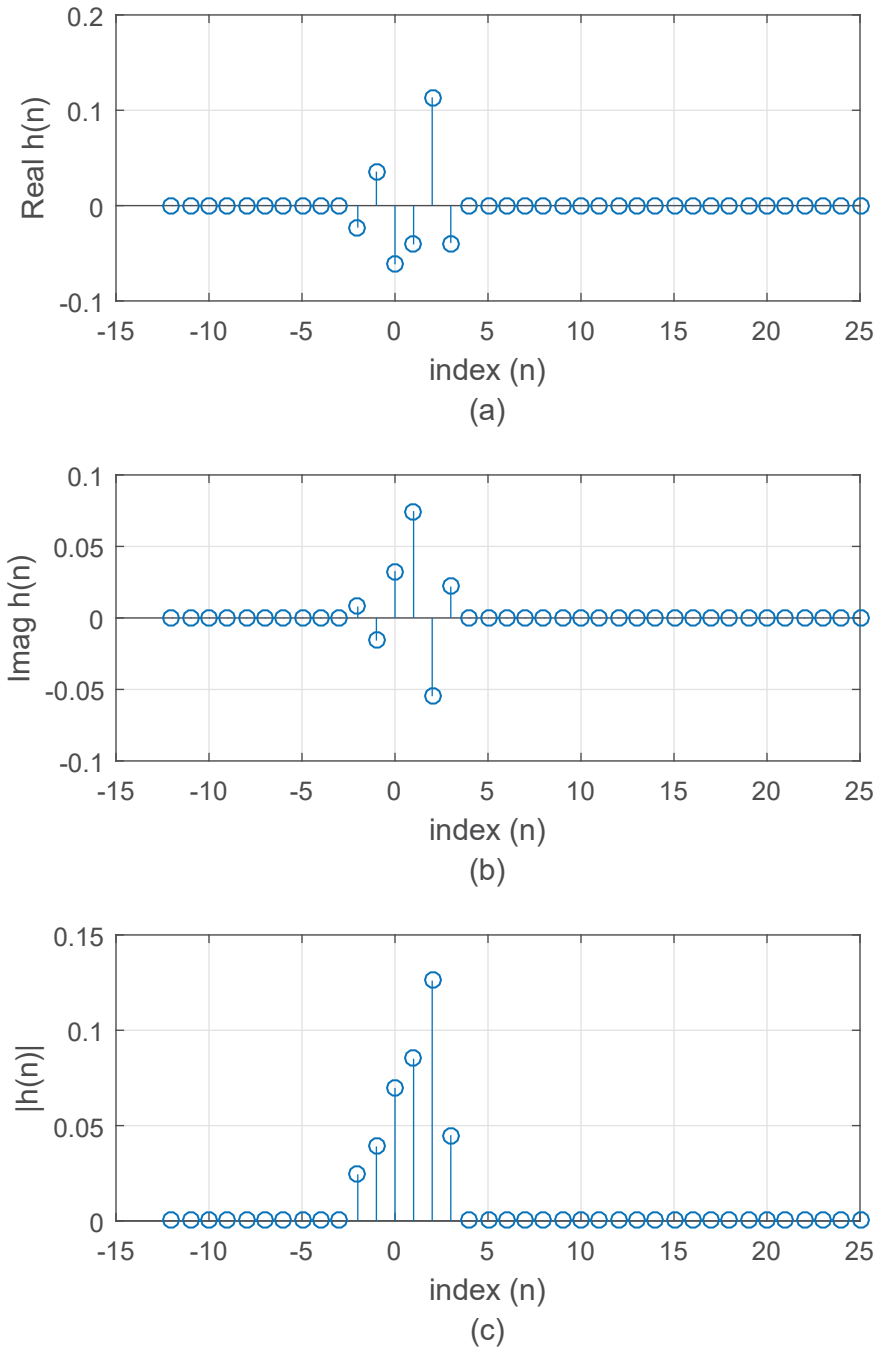


Figure 4.10: The impulse response of the channel measured on the Final Approach/Landing on Runway 22L at Edwards AFB (a) is a graph of the real coefficients; (b) is a graph of the imaginary coefficients; (c) is a graph of the magnitude of the coefficients.

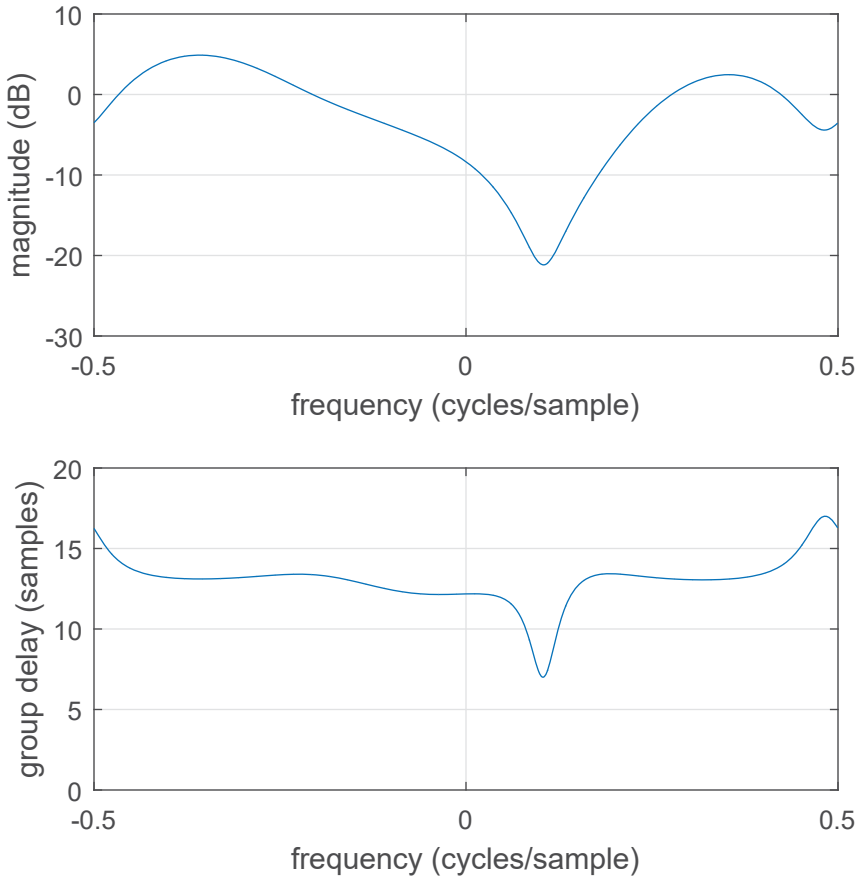


Figure 4.11: (a) is the magnitude of the frequency response of channel 4 in dB; (b) is the group delay of the channel.

output of the detection filter is downsampled to one sample per bit then rotated to compensate for the frequency offset. The rotated samples are denoted $x_r(k)$ and are used to make decisions. The phase error is computed by the phase error detector (PED) and is given by

$$e(k) = \begin{cases} d(k-1) \times \text{Im}\left\{x_r(k-1)\right\} - d(k) \times \text{Re}\left\{x_r(k)\right\} & k \text{ odd} \\ 0 & k \text{ even} \end{cases} \tag{4.2}$$

where $d(k)$ is the antipodal version of the pilot bit for

$$0 \leq k \leq L_p - 1, \tag{4.3}$$

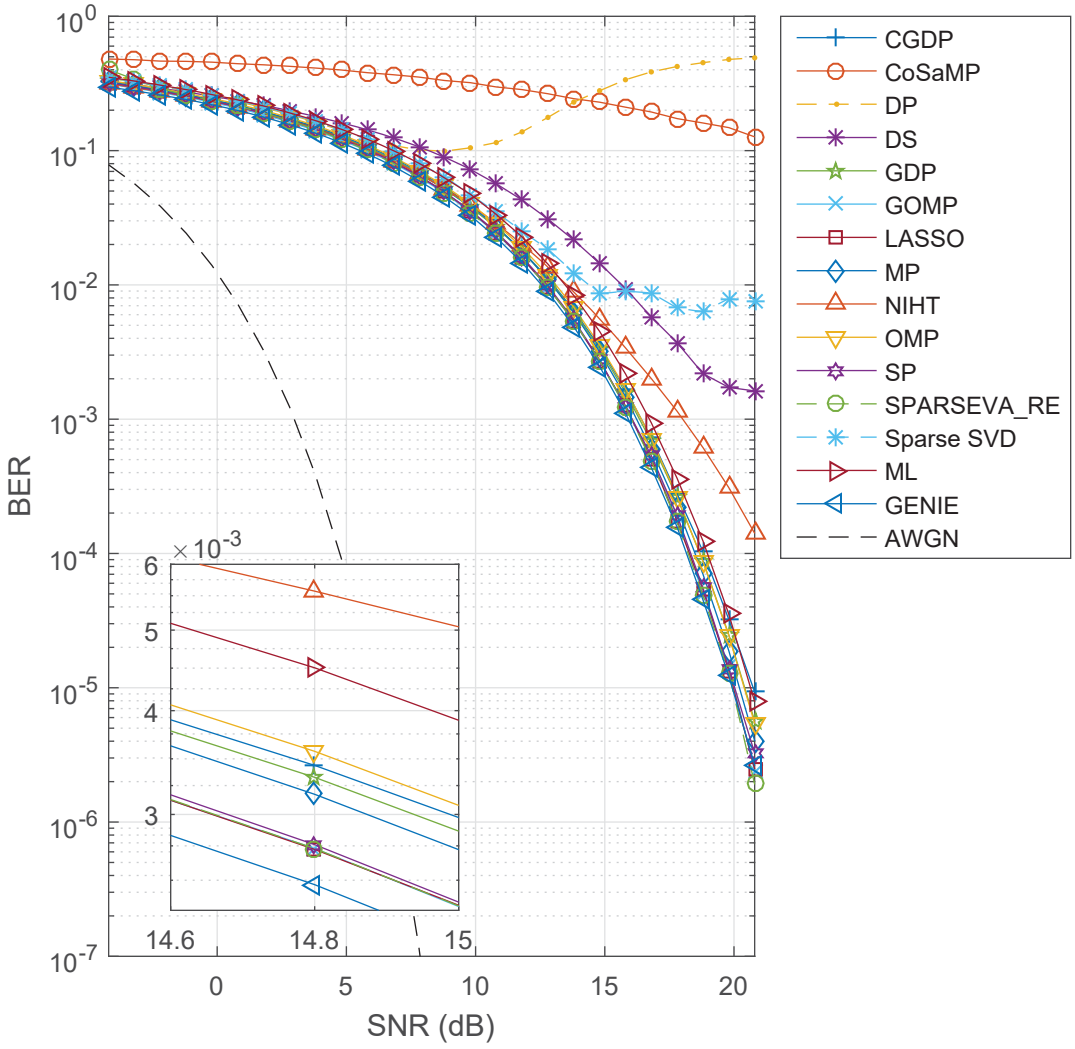


Figure 4.12: The BER curves for each of the algorithms when equalizing the channel measured on the Final Approach/Landing on Runway 22L at Edwards AFB (Channel 4).

and the output of the decision block for

$$L_p \leq k \leq L_p + L_d - 1. \tag{4.4}$$

The error signal is filtered and converted to a pair of quadratrue sinusoids using the direct digital synthesizer (DDS).

The frequency offset is found using the simple delay-and-multiply estimator described in [27]. Because the frequency estimate is not perfect, the derotation by the estimate leaves a residual

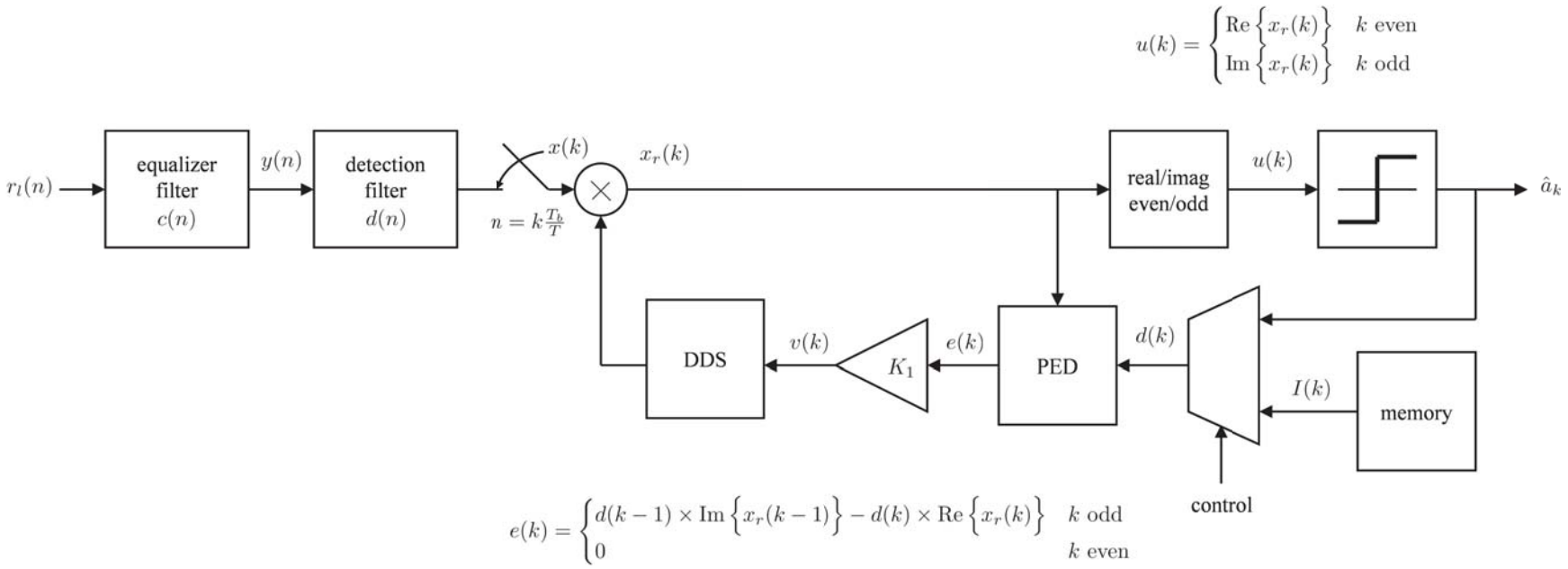


Figure 4.13: Detector that assumes perfect timing with a small carrier phase frequency offset.

frequency offset in $x_r(k)$. The frequency offset is estimated for every packet, thus a different random residual frequency offset is present in every packet. The residual frequency offset degrades the estimate produced by the ML channel estimator. In the following sections, the impact of the residual frequency offset on the sparse channel estimators is explored.

The following subsections present the BER results in the presence of a residual frequency offset introduced by the frequency offset estimator. Only the algorithms that performed well for each channel are included, thus the CoSaMP, NIHT, Sparse SVD, DP, and DS algorithms are not included.

4.2.1 Channel 1 with Frequency Offset (Taxiway E, Edwards AFB)

The BER results for Channel 1 with a residual frequency offset are illustrated in Figure 4.14. The results are similar to those shown in Figure 4.3 (Channel 1 when no frequency offset is present) except that the curves are further to the right 1 dB. The lowest BER curves are produced by SPARSEVA-RE, GOMP, and LASSO. Note that all of the sparse estimators outperform the ML estimator.

4.2.2 Channel 2 with Frequency Offset (Black Mountain Flight Corridor, Edwards AFB)

The BER results for Channel 2 with a residual frequency offset are illustrated in Figure 4.15. As with Channel 1, the BER curves in Figure 4.15 are similar to those in Figure 4.6 except all of the curves are further to the right by about 1 dB. At first glance, it might appear that SPARSEVA-RE outperforms the GENIE. However, the Monte Carlo simulations did not use sufficient iterations to support the conclusion. The slight advantage of SPARSEVA-RE over the GENIE seemed to be consistent across all the simulation runs performed in preparing this thesis. This trend was not observed in the simulation results corresponding to the other three channels. Consequently, we speculate that Channel 2 possess some property, not shared by the other three channels, that seems to favor SPARSEVA-RE. After SPARSEVA-RE, the two subsequently best estimators are GOMP and LASSO.

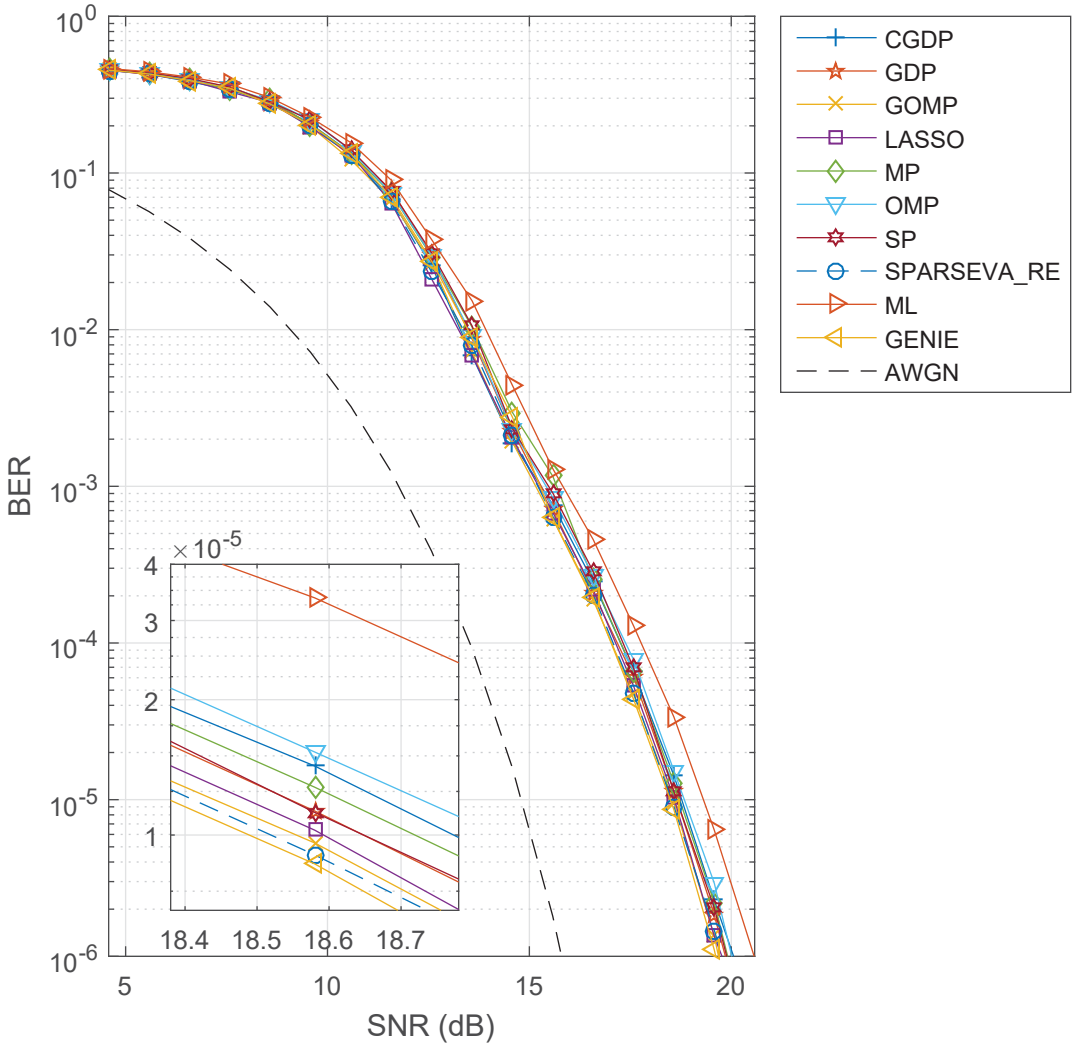


Figure 4.14: The BER curves for each of the algorithms when equalizing the channel measured on Taxiway E at Edwards AFB with a frequency offset (Channel 1).

4.2.3 Channel 3 with Frequency Offset (Cords Road Flight Corridor, Edwards AFB)

Figure 4.16 illustrates the BER curves for Channel 3 with a residual frequency offset. The results in Figure 4.16 are similar to those in Figure 4.9 (Channel 3 with no frequency offset except that the curves are further to the right by about 1 dB. The results follow the same pattern as those for Channel 1 with a frequency offset.

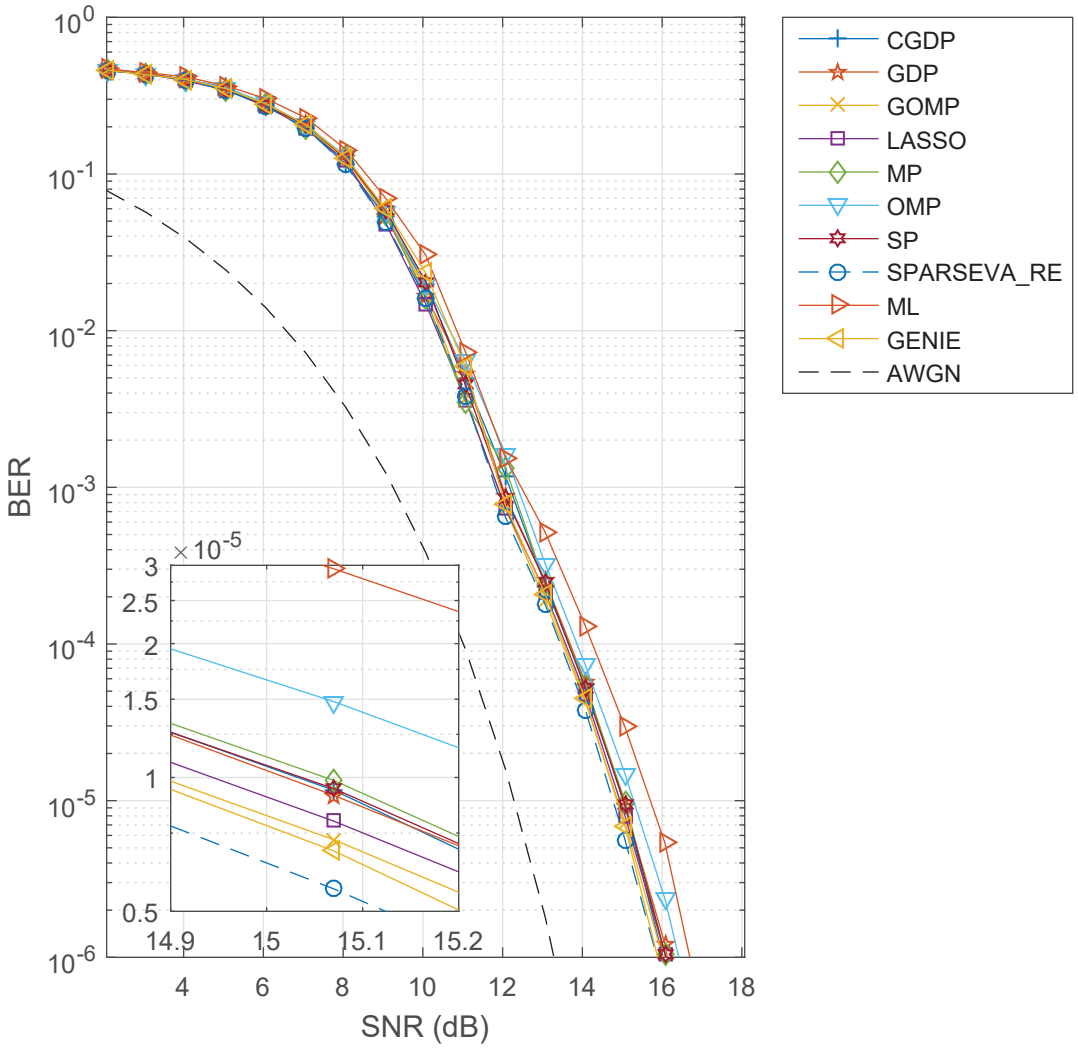


Figure 4.15: The BER curves for each successful algorithm when equalizing the channel measured on the Black Mountain flight corridor at Edwards AFB with a frequency offset (Channel 2).

4.2.4 Channel 4 with Frequency Offset (Final Approach/Landing on Runway 22L, Edwards AFB)

Figure 4.17 illustrates the BER curves for Channel 4 with a residual frequency offset. The results in Figure 4.17 are similar to those in Figure 4.12 (Channel 4 with no frequency offset) except that all of the curves are further to the right by about 1 dB. For this channel the LASSO algorithm produced the lowest BER followed by SPARSEVA-RE, and GOMP.

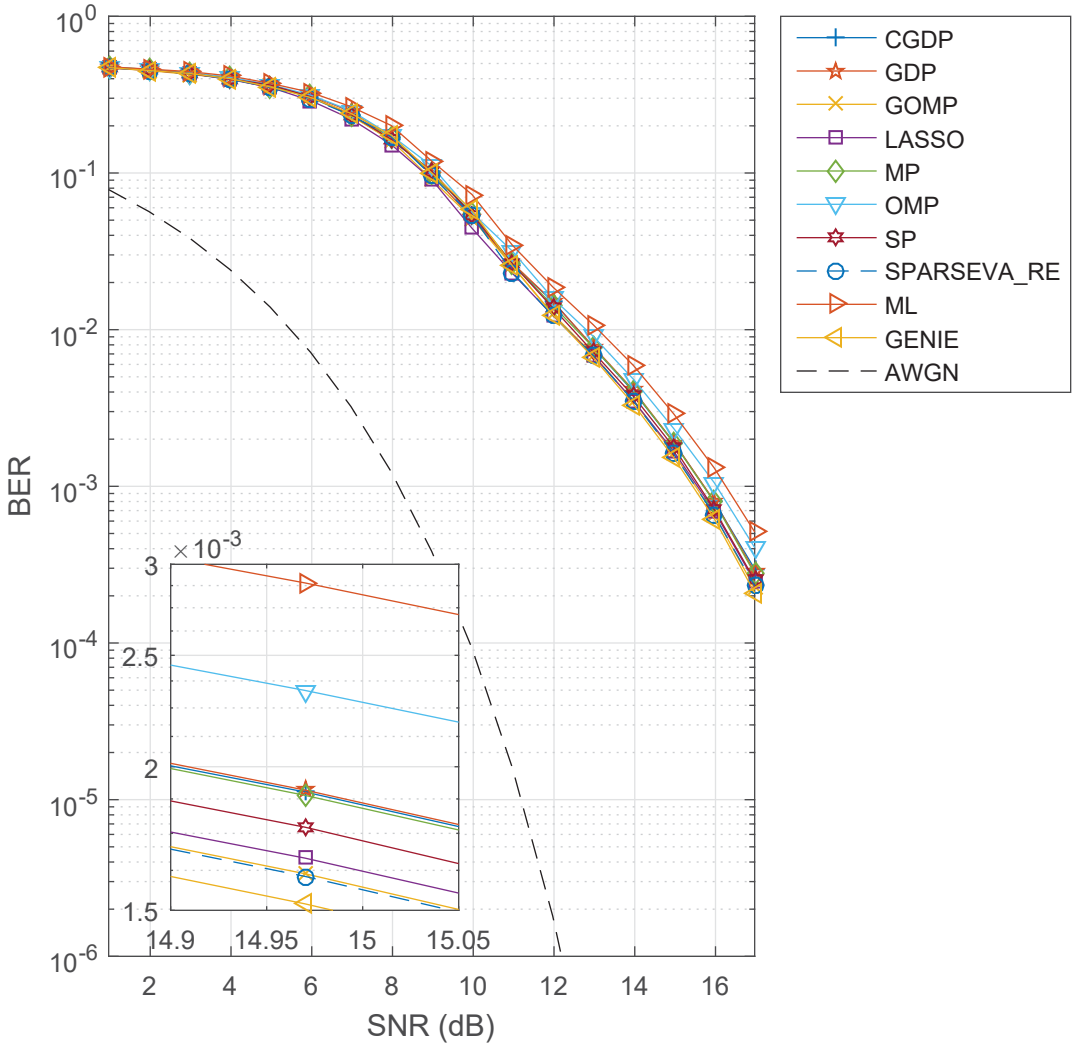


Figure 4.16: The BER curves for each successful algorithm when equalizing the channel measured on the Cords Road flight corridor at Edwards AFB with a frequency offset (Channel 3).

4.3 Summary of Results

When no residual frequency offset is present, the GOMP estimator performs the best, achieving a 1 dB gain over the ML estimator. The second and third best estimators are SPARSEVA-RE and LASSO. Most of the greedy pursuit estimators outperform the ML estimator.

When a residual frequency offset is present, the SPARSEVA-RE estimator is typically the best followed by GOMP and LASSO. The greedy pursuit estimators also outperform the ML estimator in this case.

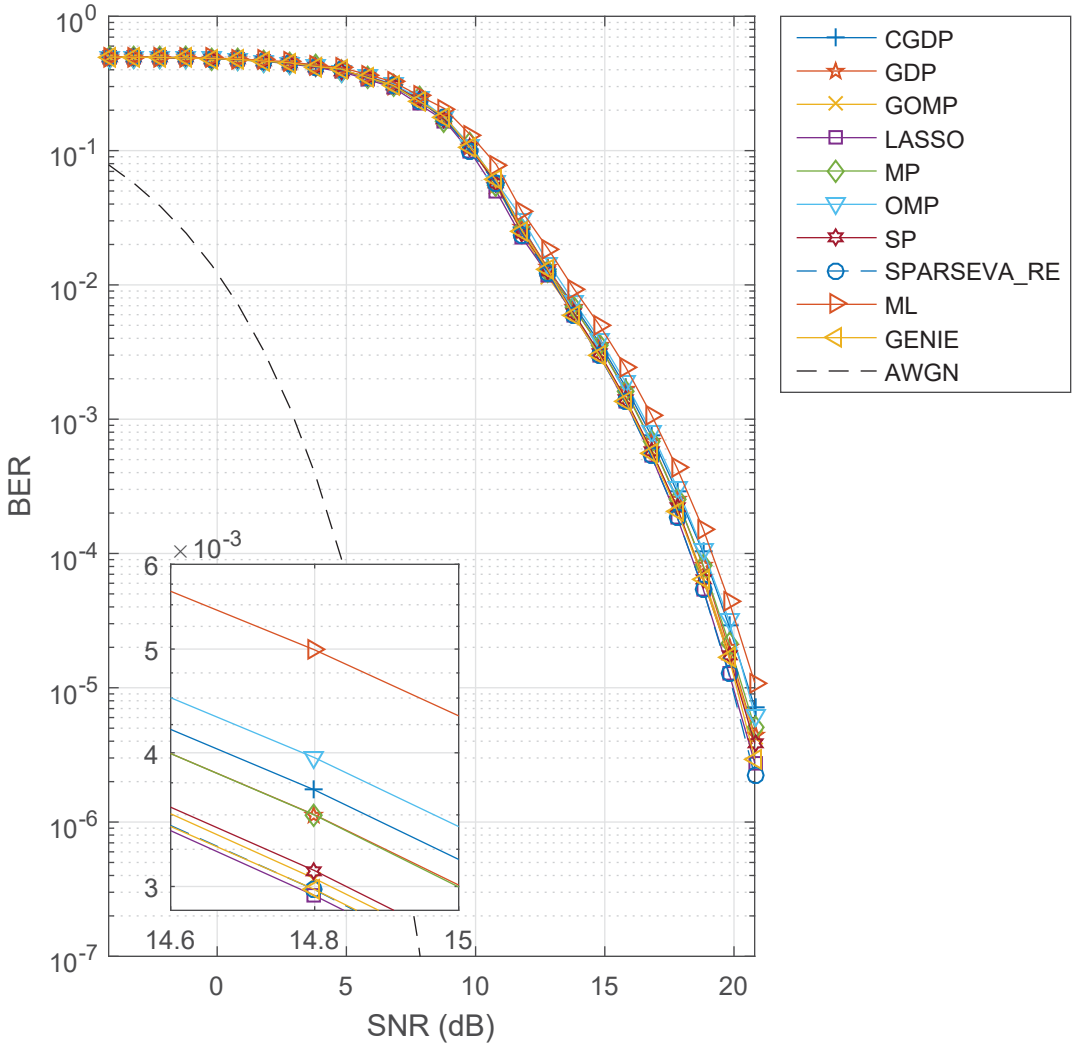


Figure 4.17: The BER curves for each successful algorithm when equalizing the channel measured on the Final Approach/Landing on Runway 22L at Edwards AFB with a frequency offset (Channel 4).

The NIHT, DS, SVD, DP, and CoSaMP did not show consistently good performance in the case of no frequency offset. For this reason, the equalized BER performance corresponding to these five algorithms was not evaluated in computer simulation. The NIHT algorithm does not consistently outperform the ML estimator and is more sensitive to the true number of non-zero channel coefficients. The performance of the DS is very sensitive to the channel it is estimating. For some channels it is a better estimator than the ML estimator and for others it is worse. The sparse SVD estimation algorithm occasionally produces a bad channel estimate and this limits the

performance of an equalizer based on its estimate. The DP algorithm does not choose the threshold λ dynamically, which results in poor performance as the SNR varies. Finally, the CoSaMP algorithm performs the worst for all SNR values and for all channels.

CHAPTER 5. CONCLUSION

5.1 Summary

This thesis surveyed a variety of sparse estimation algorithms and their application to aeronautical telemetry. The system under consideration used SOQPSK-TG modulation described in A.1 with a data structure described in A.2. The equalizer used in the system was the MMSE equalizer operating on the samples of the received waveform.

It has been shown that many of the algorithms provide a better channel estimate than the ML estimate. This has been verified using post-equalizer BERs. Of the surveyed algorithms, GOMP performed the best followed by SPARSEVA-RE and LASSO for no frequency offset between the transmitter and receiver. When a frequency offset is present, SPARSEVA-RE performs the best followed by GOMP and LASSO. The NIHT, DS, SVD, DP, and CoSaMP did not show consistently good performance.

5.2 Further Work

Additional research includes varying the length of the pilot sequence. Making the preamble short enough to create an underdetermined sensing matrix would satisfy the assumption underlying some algorithms. If similar results can be achieved with a shorter pilot sequence, then the overhead of the system decreases, allowing for improved spectral efficiency. Another interesting study would be the generation of the preamble sequence. The system used the pilot sequence defined by the iNET standard. Because the pilot sequence was not generated by a random process it is not possible to claim that the sensing matrix formed from the waveform samples corresponding to the pilot sequence satisfies the RIP conditions with a high probability. By generating the pilot from a random process, the sparse estimation algorithms may perform better as the sensing matrix

would satisfy RIP with high probability. Finally, if a reliable estimate of the number of non-zero coefficients k is available, the post-equalizer BER performance corresponding to the greedy pursuit algorithms would improve. The post-equalizer BER performance corresponding to the NIHT algorithm would show the biggest improvement as it is most susceptible to the number of non-zero coefficients to be estimated.

REFERENCES

[1] L. L. Scharf, *Statistical Signal Processing*. Addison-Wesley Reading, MA, 1991, vol. 98. 3, 16

[2] W. U. Bajwa, J. Haupt, A. M. Sayeed, and R. Nowak, "Compressed channel sensing: A new approach to estimating sparse multipath channels," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1058–1076, 2010. 3, 13, 25

[3] J. Proakis and M. Salehi, *Digital Communications*, 5th ed. New York, NY: McGraw Hill, 2008. 8

[4] F. T. Ulaby and A. E. Yagle, *Engineering Signals and Systems*. National Technology and Science Press, 2013. 9

[5] T. Moon and W. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Upper Saddle River, NJ: Prentice-Hall, 2000. 11

[6] M. Rice, M. Saquib, A. Cole-Rhodes, F. Moazzami, and E. Perrins, "Phase 1 final report: Preamble assisted equalization for aeronautical telemetry (PAQ)," 2014. 11, 55

[7] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012. 12, 13, 33, 43, 46

[8] A. Bruckstein, D. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 64–81, 2009. 12, 13

[9] L. Dai and K. Pelckmans, "Sparse estimation from noisy observations of an overdetermined linear system," *Automatica*, vol. 50, no. 11, pp. 2845–2851, 2014. 12, 13, 24, 58

[10] E. Candes and T. Tao, "The Dantzig selector: Statistical estimation when p is much larger than n," *The Annals of Statistics*, pp. 2313–2351, 2007. 13, 14, 25

[11] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996. 13, 18

[12] J. Wen, Z. Zhou, D. Li, and X. Tang, "A novel sufficient condition for generalized orthogonal matching pursuit," *IEEE Communications Letters*, 2016. 13, 41

[13] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009. 13, 15, 51

[14] C. R. Rojas and H. Hjalmarsson, "Sparse estimation based on a validation criterion," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 2825–2830. 13, 22

[15] H. Zou, "The adaptive lasso and its oracle properties," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, 2006. 13, 21

[16] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009. 13, 47

[17] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 629–654, 2008. 13, 44

[18] ——, "Normalized iterative hard thresholding: Guaranteed stability and performance," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 298–309, 2010. 13, 46, 47

[19] ——, "Gradient pursuits," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2370–2382, 2008. 13, 38, 40

[20] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993. 13, 14, 33

[21] Y. C. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*. IEEE, 1993, pp. 40–44. 13, 15, 36

[22] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005. 16

[23] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008. 16

[24] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*. Springer Series in Statistics Springer, Berlin, 2001, vol. 1. 19

[25] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, 2009. 55

[26] J.Ravert and M.Rice, "On frequency offset compensation for equalized SOQPSK," in *Proceedings of the International Telemetering Conference*, Glendale, AZ, 2016. 65

[27] M. Rice and E. Perrins, "On frequency offset estimation using the inet preamble in frequency selective fading," in *Military Communications Conference (MILCOM), 2014 IEEE*. IEEE, 2014, pp. 706–711. 68

[28] T. Nelson, E. Perrins, and M. Rice, "Near optimal common detection techniques for shaped offset QPSK and Feher's QPSK," *IEEE Transactions on Communications*, vol. 56, no. 5, pp. 724–735, 2008. 82

[29] *Radio Access Network (RAN) Standard, Version 0.7.9*, Integrated Network Enhanced Telemetry (iNET) Radio Access Network Standars Working Group, (Available on-line at https://www.tena-sda.org/display/INET/iNET+Platform+Interface+Standards). 83

[30] 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, 2016. 83

APPENDIX A. SOQPSK-TG

A.1 SOQPSK-TG Modulation

The system uses Shaped Offset Quaternary Phase Shift Keying (SOQPSK), Telemetry Group (TG) version, modulation. SOQPSK-TG is a constrained ternary partial response continuous phase modulation (CPM) waveform. The complex valued lowpass equivalent is

$$s(t) = e^{j\Phi(t;\mathbf{x})}, \tag{A.1}$$

where

$$\Phi(t;\mathbf{x}) = 2\pi h \sum_{k=0}^{n} x_n q(t - nT_b). \tag{A.2}$$

Here T_b is the bit time, $\mathbf{x} = x_0, x_1, \ldots, x_n$ where $x_n \in \{-1, 0, 1\}$ represents the data symbols, h is the modulation index, and $q(t)$ is the phase pulse defined by

$$q(t) = \int_0^t g(x)dx, \tag{A.3}$$

where $g(x)$ is the frequency pulse defined by

$$g(t) = C\frac{\cos(\frac{\pi\rho Bt}{2T_b})\sin(\frac{\pi Bt}{2T_b})}{1 - 4(\frac{\rho Bt}{2T_b})^2(\frac{\pi Bt}{2T_b})}w(t) \tag{A.4}$$

for

$$w(t) = \begin{cases} 1, & 0 \le |\frac{t}{2T_b}| \le T_1 \\ \frac{1}{2} + \frac{1}{2}\cos(\frac{\pi}{T_2}(\frac{t}{2T_b} - T_1)), & T_1 \le |\frac{t}{2T_b}| \le T_1 + T_2 \\ 0, & T_1 + T_2 < |\frac{t}{2T_b}| \end{cases}. \tag{A.5}$$

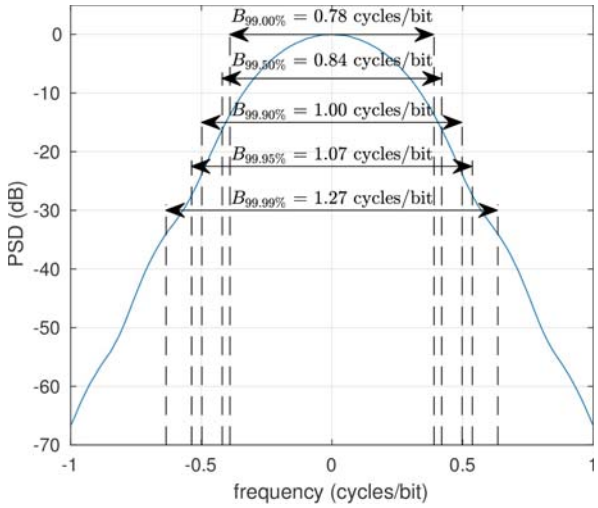


Figure A.1: Power spectral density of SOQPSK-TG waveform.

For SOQPSK-TG $h = 1/2$, $\rho = 0.7$, $B = 1.25$, $T_1 = 1.5$, and $T_2 = 0.5$ [28]. The constant C is chosen to make $g(t) = 1/2$ for $t \geq 2(T_1 + t_2)T_b$. Note that the specified values of ρ, B, T_1, and T_2 make SOQPSK-TG a partial response waveform spanning $L = 8$ bit intervals [28]. The data symbols x_k are related to the binary inputs $a_n \in \{-1, 1\}$ by

$$x_k = (-1)^{k+1} \frac{a_{k-1}(a_k - a_{k-2})}{2}. \tag{A.6}$$

The power spectral density of the SOQPSK-TG pulse shape is shown in Figure A.1. Notice that 99.9% of the power is contained in the interval -0.5 to 0.5 cycles/bit (or -0.25 to 0.25 cycles/sample at two samples per bit).

A.2 Pilot Sequence

Data is sent with periodically inserted pilot sequences defined by the integrated Network Enhanced Telemetry (iNET) [29]. The pilot sequence is a 192-bits long and is inserted every 6144 bits. The data structure is shown in Figure A.2.

The system under consideration uses SOQPSK-TG modulation defined in Appendix A.1. Due to the non-linear relationship between the symbols and the waveform, the samples corresponding to the pilot sequence are known by the receiver instead of the symbols. The location of the pilot is assumed to be known and curious reader is reffered to [30] for more information on pilot detection.

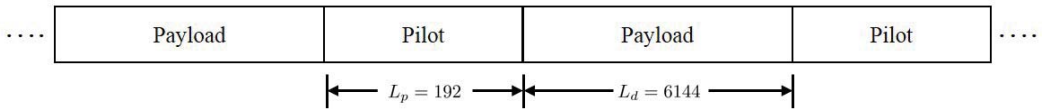


Figure A.2: Data structure.