NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DOPPLER CORRECTION FOR DIGITAL MODULATION IN A FADING CHANNEL WITH SISO

by

Gabriel Tham Chi Mun

December 2010

Thesis Co-Advisors: Tri Ha
Su Weilian

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2010	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Doppler Correction for Digital Modulation in a Fading Channel With SISO		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Gabriel Tham Chi Mun			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number N/A..			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

In a high mobility wireless channel, the Doppler effect is compounded and must be corrected by using pilot-aided symbols. The pilot symbol rate depends on the severity of the Doppler effect. There are existing algorithms such as differential decision-feedback (D-DF) and double-differential decision-feedback (DD-DF) for single-input single-output (SISO) systems to improve channel tap estimation. Such algorithms improve the bit error rate (BER) performance of pilot symbol aided decision feedback demodulation. In this thesis, the use of minimum mean square error (MMSE) estimation was implemented to further improve channel tap estimation for the D-DF and DD-DF algorithms.

BER performance showed significant improvement for higher order modulation schemes. On the other hand, implementation of the new algorithm on quadrature phase-shift keying (QPSK) showed negligible improvement. Also, the MMSE algorithm is not very effective for cases with very high Doppler frequency shifts.

Finally, it was observed that the performance improvement due to MMSE estimation decreases as the signal-to-noise ratio increases. To counter this, adaptive tolerances, tied to the variance of channel tap estimation, were found to provide better channel estimation surge detection capabilities. Such implementation dramatically improved the bit error rate performances for D-DF with the MMSE algorithm but was not effective for the DD-DF algorithm.

14. SUBJECT TERMS QPSK, QAM, Fading Channel, Doppler Effect, Minimum Mean Square Error, Pilot Symbol-Aided Demodulation			**15. NUMBER OF PAGES** 101
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DOPPLER CORRECTION FOR DIGITAL MODULATION IN A FADING CHANNEL WITH SINGLE-INPUT SINGLE-OUTPUT

Gabriel Tham Chi Mun
Military Expert 5, Republic of Singapore Air Force
B.S., National University of Singapore, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2010

Author: Gabriel Tham Chi Mun

Approved by: Tri Ha
Thesis Co-Advisor

Su Weilian
Thesis Co-Advisor

R. Clark Robertson
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In a high mobility wireless channel, the Doppler effect is compounded and must be corrected by using pilot-aided symbols. The pilot symbol rate depends on the severity of the Doppler effect. There are existing algorithms such as differential decision-feedback (D-DF) and double-differential decision-feedback (DD-DF) for single-input single-output (SISO) systems to improve channel tap estimation. Such algorithms improve the bit error rate (BER) performance of pilot symbol aided decision feedback demodulation. In this thesis, the use of minimum mean square error (MMSE) estimation was implemented to further improve channel tap estimation for the D-DF and DD-DF algorithms.

BER performance showed significant improvement for higher order modulation schemes. On the other hand, implementation of the new algorithm on quadrature phase-shift keying (QPSK) showed negligible improvement. Also, the MMSE algorithm is not very effective for cases with very high Doppler frequency shifts.

Finally, it was observed that the performance improvement due to MMSE estimation decreases as the signal-to-noise ratio increases. To counter this, adaptive tolerances, tied to the variance of channel tap estimation, were found to provide better channel estimation surge detection capabilities. Such implementation dramatically improved the bit error rate performances for D-DF with the MMSE algorithm but was not effective for the DD-DF algorithm.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AWGN	Additive White Gaussian Noise
BER	Bit Error Ratio
D-DF	Differential-Decision Feedback
DD-DF	Double-Differential Decision-Feedback
LTE	Long Term Evolution
MMSE	Minimum Mean Square Error
OFDM	Orthogonal Frequency Division Multiplexing
PSK	Phase-Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
SISO	Single-Input Single-Output
SNR	Signal-to-Noise Ratio
SPS	Symbols per Second
WiMAX	Worldwide Interoperability for Microwave Access
WSS	Wide-Sense Stationary

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

In a high mobility wireless channel, the Doppler effect is compounded and must be corrected by using pilot-aided symbols. The pilot symbol rate depends on the severity of the Doppler effect. There are existing algorithms such as differential decision-feedback (D-DF) and double-differential decision-feedback (DD-DF) for single-input single-output (SISO) systems to improve channel tap estimation. Such algorithms improve the bit error rate (BER) performance of pilot symbol aided decision feedback demodulation. The thesis on "Doppler Effects on Single-Carrier Signals Operating in Fading Channel" presented the performance of differential decision-feedback and double-differential decision-feedback algorithms in fading channels for quadrature phase-shift keying (QPSK), 16-quadrature amplitude modulation (16-QAM) and 64-quadrature amplitude modulation (64-QAM) . In this thesis, the use of minimum mean square error (MMSE) estimation was implemented to further improve the channel tap estimation for differential and double-differential decision-feedback algorithms. The implementation of MMSE estimation requires statistical information on the channel tap and these are obtained from the estimates provided by the D-DF and DD-DF algorithms.

BER showed significant improvement for higher order modulation schemes such as 16-QAM, 64-QAM and 256-QAM. On the other hand, implementation of the new algorithm on modulation schemes like QPSK showed negligible improvement. The MMSE algorithm proved to be not effective for cases with very high Doppler frequency shifts.

Finally, it was observed that the performance improvement due to MMSE estimation decreases as signal-to-noise ratio increases. To counter this, adaptive tolerances were used to detect large changes in channel estimation. Since the channel is expected to be slow fading within the pilot symbol frame, the adaptive tolerances, tied to the variance of channel tap estimation, were found to provide better surge detection capabilities. Such implementation dramatically improved the BER performances for the D-DF with MMSE algorithm but was not effective for the DD-DF algorithm.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

It is both a privilege and honor to be under the guidance of my thesis advisors, Professor Tri Ha and Professor Su Weilian, who have taught me both the fundamentals of digital communications and how to conduct my research in search of new knowledge.

I would also like to extend my gratitude to the faculty, laboratory and administrative staff in the Department of Electrical and Computer Engineering for their generous instruction and support.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OVERVIEW

Digital communication has always been challenging in a high mobility fading environment. On top of the typical additive white Gaussian noise (AWGN) model, the realistic environment complicates the channel model by introducing multi-path effects. The multiple paths create varying magnitude and phase over time and space. The high mobility scenario creates channel that vary quickly over time. This makes demodulation very difficult. Although the varying magnitude does cause severe attenuation at times, known as deep fades, such effects are temporal and cause relatively low bit errors. The key contribution to the bit errors are phase errors that could shift the data information phase dramatically over time and cause erroneous demodulation. In order to compensate for the phase changes, pilot symbols are embedded in the data stream for channel estimation. If the pilot symbol rate is lower than the coherence time, the channel would be fading slowly between two pilot symbols. Within this frame, the channel tap phase error could be assumed to vary slowly. Even with the simple use of the channel tap phase error from the pilot symbol, the bit error rate performance is still very poor. Therefore, there is a need for better channel estimation techniques.

One such technique is the use of differential decision-feedback (D-DF) and double-differential decision-feedback (DD-DF) algorithms [1]. The differential and double-differential decision-feedback algorithms use the channel tap phase error estimate from previous symbols rather than from the pilot symbol. After determining the latest channel tap phase estimate from the decoded symbol, assuming that there is no decoding error, the channel tap estimate is updated for the next symbol. This allows for a more accurate estimate, especially when the data symbol is spaced far in time from the pilot symbol.

B. OBJECTIVE

In this thesis, the key objective is to study the feasibility of using minimum mean square error on the differential and double-differential decision-feedback algorithms to improve the overall bit error rate performance of single-input single-output (SISO) communications in high mobility fading environments. Simulation was carried out to study the performances obtained with and without minimum mean square error estimation. Simulations on modulations such as QPSK, 16-QAM, 64-QAM and 256-QAM were developed in MATLAB and used as a baseline for comparison. Once the minimum mean square error estimate is incorporated, new algorithms to further enhance the bit error rate performances could be developed.

C. OUTLINE

This thesis consists of four main chapters. The first chapter provides an overview of the impetus and objectives of the thesis, while Chapter II begins the theoretical review of fading channels and pilot symbol-aided decision feedback demodulation. This chapter also reviews previous work on differential and double-differential decision-feedback algorithms. Chapter III begins with the study of minimum mean square error and the orthogonality principle and proceeds to discuss the implementation of minimum mean square error onto differential and double-differential decision-feedback algorithms. The new algorithms were further enhanced with adaptive tolerance, which made use of channel tap estimate variance to identify surges in channel tap estimates probably due to erroneous demodulation.

II. REVIEW OF PREVIOUS WORK

A. FADING CHANNEL AND PILOT SYMBOL DECISION FEEDBACK DEMODULATION

1. Fading Channel Overview

In a multipath environment, the transmitted signal travels through several different paths before arriving at the receiver. The different paths arise from the signal reflecting and refracting around obstacles in its own respective paths. This gives rise to several copies of the original signal with different delay, phase shift and attenuation. These multipath copies combine at the receiver with a combined signal envelope that can either be modeled as a Raleigh, Rician or Nagakami fading process and a phase shift of uniform distribution. The types of fading processes are defined by different conditions such as the presence of the line-of-sight path. When there is a direct line-of-sight between the transmitter and the receiver, the fading process can be modeled as Rician fading. On the other hand, a Rayleigh fading process results from non-line-of-sight communication and is one of the worst types of fading. The amplitude of such fading is defined by the density function with normalized channel tap such that $\mathrm{E}\left(|h|^2\right)=1$ where h is the complex path attenuation, often referred to as the channel tap,

$$f_{|h|}(x)=2xe^{-x^2}. \tag{1}$$

The fading channel effect is defined in terms of time-varying and space-varying effects. The space-varying effect arises from the multipath effect and can be measured by the multipath delay spread of the fading channel, denoted by τ_d. The multipath delay spread is defined as the difference between the maximum and minimum time delay. The coherence bandwidth B_c is computed as the inverse of the multipath delay spread. When the coherence bandwidth is larger than that of the signal, the fading process is defined as frequency-selective fading.

The time-varying effect arises mainly from the Doppler frequency shift f_d due to motion of the transmitter, receiver or the obstacles within the path of propagation. This time-varying effect affects both the amplitude and phase shift of the received signal. The phase shift affects the channel estimation interval, also known as pilot symbol interval. Using the pilot symbols, we can correct the aggregated phase shift. A good indication of this time-varying effect is the coherence time T_c which is related to the Doppler spread f_D. The Doppler spread is defined as the difference between the maximum of both positive Doppler shifts and negative Doppler shifts, and the coherence time T_c is calculated from

$$T_c = \frac{1}{4 f_D}. \tag{2}$$

The coherence bandwidth is calculated using the formula $B_c = 1/\tau_d$, where τ_d is the maximum channel path delay. In summary, the fading channel can then be classified based on the coherence time and coherence bandwidth. When the coherence time T_c is shorter than the channel estimation interval, the channel is defined as slow fading, relatively unchanging within the channel estimation interval. Otherwise, it is considered as fast fading. On the other hand, if the coherence bandwidth is larger than the signal bandwidth, the channel is flat, or frequency non-selective, fading. This case is also applicable for scenarios where there is only one resolvable path. [1]

2. Clark-Doppler Power Spectrum

The multipath autocorrelation and Doppler profiles are essential in the understanding of a fading channel and can be determined from the channel impulse response. The overall channel impulse response is comprised of several complex channel taps, denoted by $h(\tau, t)$, where τ is the path delay and t is the time to which the channel varies. Replacing the channel impulse response with the summation of the i^{th} resolvable paths $\sum_i h_i(t)\delta(\tau - \tau_i(t))$, the autocorrelation function is given by

$$\begin{aligned} R_h(\tau,\tau';t_1,t_2) &= \mathrm{E}\left[h(\tau,t_1)h^*(\tau',t_2)\right] \\ &= \sum_i \mathrm{E}\left[h_i(t_1)h_i^*(t_2)\right]\delta\left(\tau-\tau_i(t_1)\right). \end{aligned} \tag{3}$$

Given that the channel can be modeled as a wide-sense stationary (WSS) process and is ergodic, as well as assuming flat fading and only one i^{th} resolvable path, the autocorrelation simplifies to

$$R_h(t_2-t_1) = \mathrm{E}\left[h_i(t_1)h_i^*(t_2)\right]. \tag{4}$$

The Clark-Doppler power spectrum can be derived under the assumption that the fading channel is flat and that the paths that arrive at the receiver are in a single plane. If there is only one resolvable path, the Clark-Doppler power spectrum provides the autocorrelation function of a fading channel. The multipath autocorrelation function for random horizontal arrival angles is given as

$$R_h(t') = PJ_0(2\pi f_d t') \tag{5}$$

where $t' = t_2 - t_1$ is the time difference between any arbitrary times, t_1 and t_2, P is the total received power $\mathrm{E}\left[|h_k|^2\right]$, J_0 is the zero order Bessel function of the first kind and f_d is the Doppler frequency[1].

3. Pilot Symbol-Aided Decision Feedback Demodulation

In order to perform coherent demodulation, the phase shifts due to the fading channel must be estimated and corrected before demodulation. If the phase errors are allowed to accumulate without any correction, the decoded symbol will be in error. One common means of phase correction is through the use of pilot symbols. With the proper selection of pilot symbols and their intervals, the fading channel can be estimated. This channel tap can then be used to correct for any phase shift caused by the channel. The pilot symbol interval should be less than the coherence time in order for the channel to be relatively static, or slow fading. On the other hand, the pilot symbol rate must also be much smaller than the data rate to maximize data throughput. The channel estimation from the pilot symbol $h_p = |h_p|e^{j\theta_p}$, where $|h_p|$ is the attenuation and θ_p is the phase shift due to the channel for the pilot symbol, can be used to correct the data symbol until the next pilot symbol is transmitted. The bit error rate performances of different modulation

schemes with pilot symbol-aided decision feedback are shown in Figure 1. The parameters for the simulations were a symbol rate of 1,000,000 symbols per second (1 Msps), a Doppler shift of 50 Hertz and a pilot symbol interval of 50 symbols.

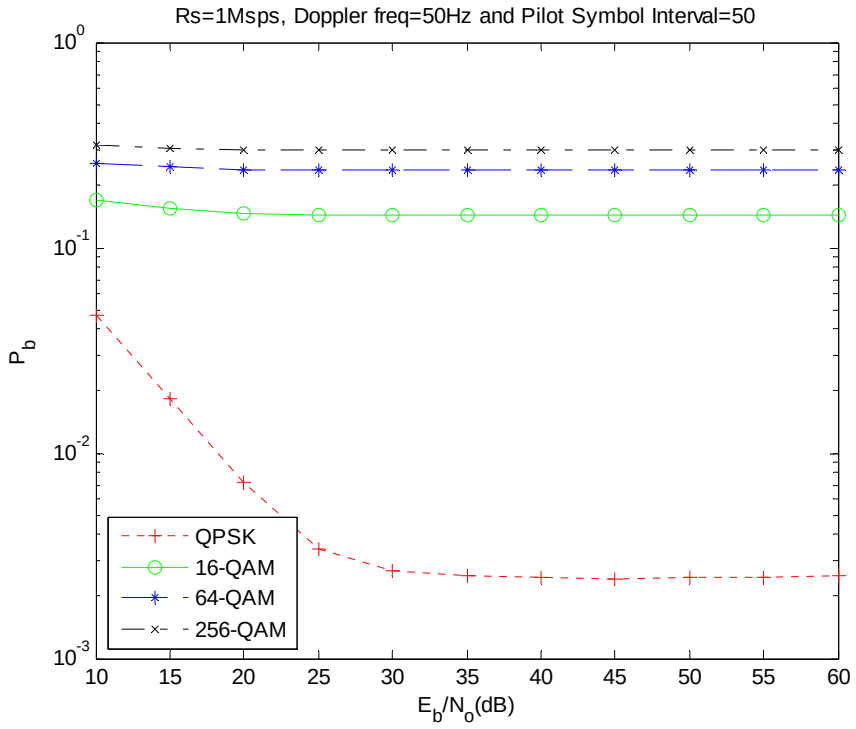


Figure 1. BER of Pilot Symbol-Aided Decision Feedback Demodulation.

B. DIFFERENTIAL DECISION-FEEDBACK (D-DF) AND DOUBLE-DIFFERENTIAL DECISION-FEEDBACK (DD-DF) ALGORITHMS

1. Differential Decision-Feedback Algorithm

The differential decision-feedback algorithm provides a more updated estimate of the channel tap phase compared to simply using the pilot symbol to estimate the channel tap phase. Still using the pilot symbols, the channel estimate can be used to correct the phase errors in the symbol following the pilot symbol. The algorithm, instead of using the same channel tap phase estimation $\hat{h}_p^* \Big/ \left|\hat{h}_p\right| = e^{-j\theta_p}$ for the entire frame of symbols until the next pilot symbol also uses the previous channel tap phase error $\hat{\varepsilon}_{l-1}$ to provide an update to the estimate. This results in lower bit error probability since the demodulation

is now dependent on the change in phase error, which is a much smaller value than the actual phase error. Given that the previous symbol is detected correctly, the previous channel tap estimate is used on the next symbol. Hence, defining the l^{th} transmitted symbol as s_l, the symbol to be decoded X_l is

$$X_l = \left(\hat{h}_p^* \Big/ \left|\hat{h}_p\right|\right)\left(\left|h_l\right| e^{j(\theta_p+\varepsilon_l)} s_l + N_l\right) e^{e^{j\hat{\varepsilon}_{l-1}}} = \left|h_l\right| e^{j\left(\varepsilon_l - \hat{\varepsilon}_{l-1}\right)} s_l + \mathrm{N}_l e^{j\hat{\varepsilon}_{l-1}} \tag{6}$$

where $h_l = \left|h_l\right| e^{j\varepsilon_l}$ is the l^{th} symbol channel tap and N_l is the complex noise for l^{th} symbol, $\mathrm{N}_l = N_l e^{-j\hat{\theta}_p}$. It can be seen from equation (6) that the decoded symbol X_l only has the phase error of $\Delta\varepsilon = \varepsilon_l - \hat{\varepsilon}_{l-1}$ after correcting with the pilot symbol's channel tap phase estimate θ_p and the phase error estimate of the *(l-1)*$^{\text{th}}$ symbol $\hat{\varepsilon}_{l-1}$.

The block diagram of the differential decision-feedback algorithm is shown in Figure 2. The differential decision-feedback algorithm can be represented by the blocks highlighted within the dotted box, which is inserted between the matched filter and the detector in a typical demodulator. The demultiplexer identifies the pilot symbol from the other symbols. The phase information from the pilot symbol and the other symbols are sent to the phase error estimator, where the channel tap phase error is estimated. The output of the phase error estimator, which is the new channel phase error estimate, is added to the pilot symbol phase error for the next symbol's phase correction. Also, the phase error estimator uses a fixed tolerance to detect spurious channel estimates. If the channel phase estimate exceeds the fixed tolerance, it is considered as erroneous, likely due to an error in decoding, and the previous channel estimate is used instead. Finally, according to Figure 2, the phase estimate from the pilot symbol and the channel tap phase estimate of the *(l-1)*$^{\text{th}}$ symbol are subtracted from the next symbol.

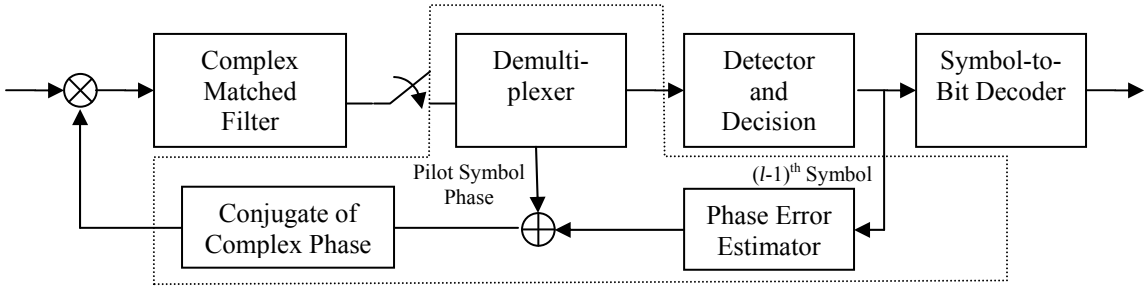


Figure 2. Block Diagram of Complex Pilot Symbol Demodulator with D-DF Algorithm.

2. Double-Differential Decision Feedback Algorithm

To further enhance the channel tap phase estimation, instead of using just the previous symbol's channel tap estimate, the channel estimation for the *(l-2)*th symbol can also be used. This double differential decision-feedback algorithm corrects the current phase by $\theta = \theta_p + 2\hat{\varepsilon}_{l-1} - \hat{\varepsilon}_{l-2}$, where $\hat{\varepsilon}_{l-1}$ and $\hat{\varepsilon}_{l-2}$ are the post-estimated differential Doppler phase error of *(l-1)*th and *(l-2)*th symbols, respectively, and θ_p is the phase shift estimated by using the pilot symbol. This approach assumes that there is a simple linear relationship between the current Doppler phase error and the past two estimated phase errors. The block diagram of the double differential decision-feedback algorithm is shown in Figure 3. The block diagram is similar to the differential decision-feedback algorithm except that the *(l-2)*th symbol is also sent to the phase error estimator to determine the channel tap phase error.

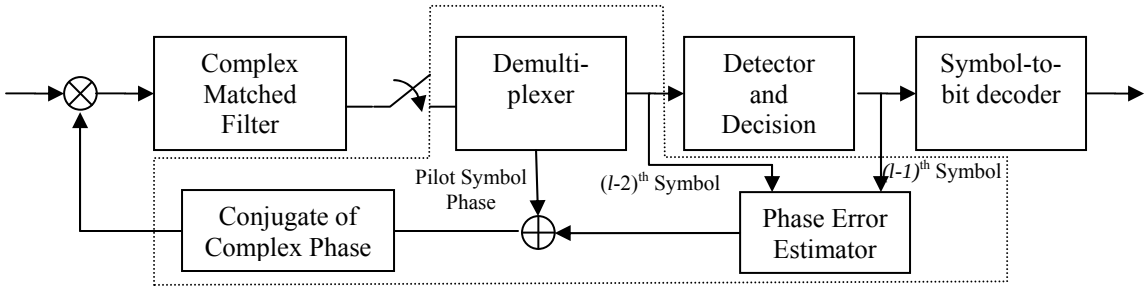


Figure 3. Block Diagram of Complex Pilot Symbol Demodulator with DD-DF Algorithm.

C. REVIEW OF PAST WORK

1. Doppler Effects on Single-Carrier Signals Operating in Fading Channel Overview

The thesis on "Doppler Effects on Single-Carrier Signals Operating in Fading Channel" [2] presents the performance of differential decision-feedback and double-differential decision-feedback algorithms in fading channels for QPSK, 16-QAM and 64-QAM modulations. Variations in pilot symbol interval, data rates and Doppler shifts were used to compare their impact on the algorithms. It was shown in the thesis that the differential decision-feedback algorithm performs remarkably well in all scenarios with the expected behavior of increasing bit error rates with increasing pilot symbol intervals and higher Doppler frequency shifts. The bit error rates with the implementation of the double-differential decision-feedback algorithm, however, showed less than expected performance, with generally similar or worse-off performance as compared to the differential decision-feedback algorithm.

2. Review of MATLAB Simulation Code and Simulation Results

a. Simulation Approach

MATLAB was used to implement the D-DF and DD-DF algorithms and validate their performance. The MATLAB communication toolbox was used to simplify the modeling of modulation, demodulation processes and the channel. A Rayleigh fading channel with additive white Gaussian noise (AWGN) was used as the model's channel. As part of the demodulation process, the differential decision-feedback and double-differential decision-feedback algorithms were also modeled. To ensure that the simulated bit error rates were accurate, sufficient data points are required. To ensure accuracy, up to 350 simulation trials of 10,000 symbols each were performed.

b. Code Changes and Improvements

After scrutinizing and reproducing the MATLAB codes provided in the thesis, some errors were found and they are listed as follows:

(1)..The number of trials does not always end with more than 350 trials. This was due to the conditions provided in the WHILE loop that determines the number of trials. While it is ideal to have a very high number of data points for accuracy, it turned out that 20,000,000 data points were generally sufficient for most cases. Hence, the code was improved to at least run 2,000 trials of 10,000 symbols each and, at the same time, the change in the resulting bit error rates after each trial was also used as a criterion for limiting the number of trials. The criterion of checking the changes in the bit error rates between two trials was put in place to increase the number of trials beyond 2,000 in the case when there were still significant changes in bit error rates. In other words, if the changes in bit error rates between two trials were still significant, the number of trials would continue after 2,000. In most cases, except for those with high E_b/N_o, the simulations ended with 2,000 trials. For those that exceeded the number of trials, some went as high as 7,000 trials.

(2) In the MATLAB code for the differential decision-feedback algorithm, the current symbol's post-estimated Doppler phase error was formulated by using only the previous symbol's channel tap. According to the algorithm, the post-estimated Doppler effect should be based on the previous symbol's estimated phase error and the pilot symbol phase. The corrected code has similar performance. The 'wrong' code used the channel tap estimation from the previous symbol and, since the fading channel is slow, the marginal differences in the channel tap are close to the correction given by the differential decision-feedback algorithm. The double differential decision-feedback algorithm coding was correct with no need for any changes. Therefore, by using the minimum mean square error approach to estimate the current symbol's channel tap, performance is expected to improve.

(3) Another observation from the MATLAB code in the thesis was that the tolerance used to identify channel tap surges resulting from decoding error had greater tolerances for the transitions between angles near the π and -π boundary. The previous conditions set for the tolerances were to compare absolute difference between the current channel tap phase error estimate with the previous estimate. However, this simple implementation does not work when the channel tap phase error estimates changes between angles near the π and -π boundary. Due to the way MATLAB calculates the angle of a complex number, when the angle is more than π, it returns a negative angle. The difference in the bit error rate performances is shown in Figure 4. It can be seen that there is a significant difference in bit error performances, especially for large E_b/N_o. The parameters for the simulations were a symbol rate of 1,000,000 symbols per second (1 Msps), a Doppler shift 50 Hertz and a pilot symbol interval of 50 symbols.

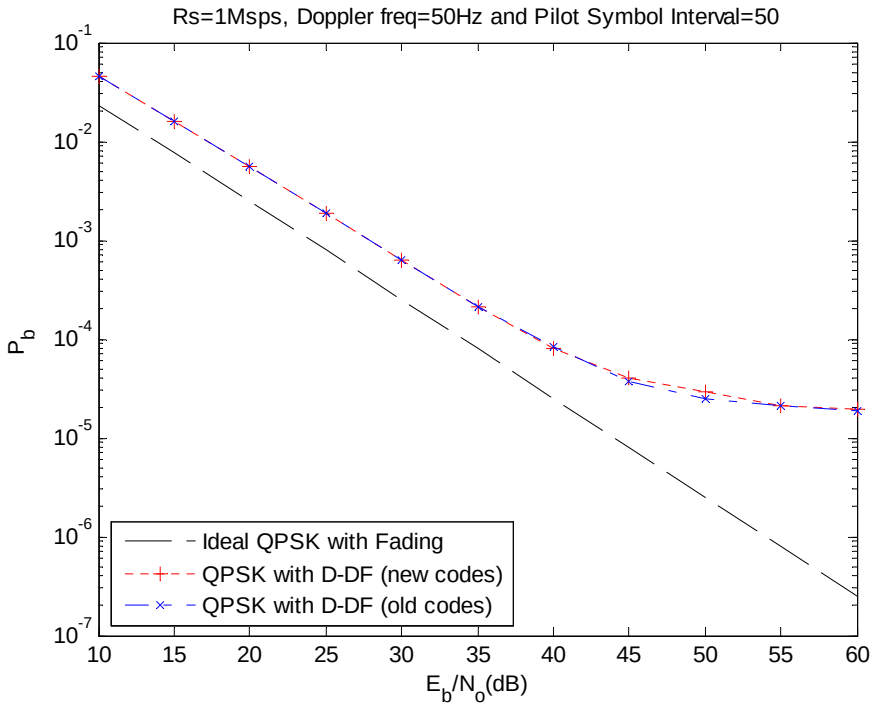


Figure 4. Comparison between Previous and Final Amended D-DF Codes.

(4) The final difference discovered was the performance between the differential decision-feedback and double-differential decision-feedback algorithms. As the number of trials increased, the differences in performance between the two algorithms narrow. This shows that there are marginal differences between the two algorithms. The double-differential decision-feedback algorithm does result in higher bit error rates at lower signal-to-noise ratios compared to the differential decision-feedback algorithms, especially for higher order modulation schemes. This is consistent with the conclusion shown in [2]. The conclusion that performance is worse for in the higher signal-to-noise ratios is only valid for certain modulation schemes, mainly those with higher order modulations. The bit error rates obtained by executing the simulation provided by the thesis updated with the changes mentioned above are shown in Figure 5. The bit error rate performances of the amended D-DF and the DD-DF codes are shown where there is an insignificant performance difference for QPSK.

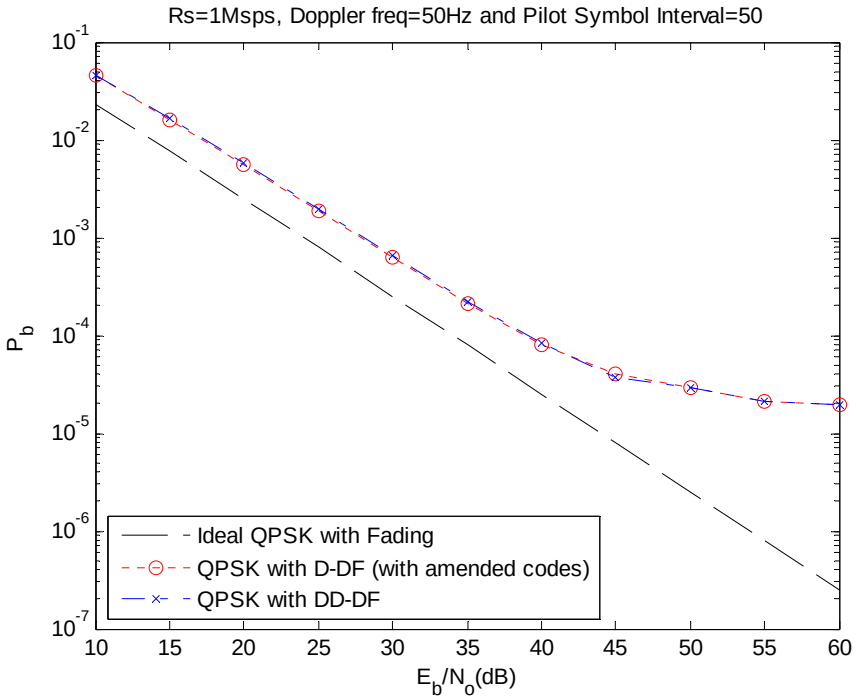


Figure 5. Comparison between Amended D-DF Codes and DD-DF.

III. DIFFERENTIAL DECISION-FEEDBACK AND DOUBLE DIFFERENTIAL DECISION-FEEDBACK WITH MMSE FOR SISO

A. MINIMUM MEAN SQUARE ERROR FILTERING

1. Introduction

With the intent of further enhancing the channel tap phase estimation, the linear minimum mean square error (MMSE) filtering approach was implemented on the phase error estimate. The concept was to use the channel estimates from the pilot symbol and subsequent estimations from the differential decision-feedback and double differential decision-feedback algorithms to obtain the actual channel tap estimate. Other than the channel tap phase estimate from the pilot symbol, the rest of the channel tap phase estimates from the other symbols would contain errors. These errors can be reduced based on the orthogonality principle. The minimum mean square error approach uses the estimates' statistical properties to better approximate the actual channel tap conditions.

2. Mean Square Signal Estimation and Orthogonality Principle

The minimum mean square approach attempts to remove noise from a signal by minimizing the expected value of the squared difference between the actual signal and the noisy signal. If the noisy signal is x and the original signal is s, the minimum mean square approach would provide an estimate $\hat{d}$ where $\mathrm{E}\left\{\left|s-\hat{d}\right|^2\right\}$ is minimized.

The orthogonality theorem states that, given estimation error $\varepsilon = s - \hat{d}$, a matrix a that minimizes the error $\mathrm{E}\left\{\left|s-\hat{d}\right|^2\right\}$ can be chosen such that $\mathrm{E}\{x_i \varepsilon\} = 0$ for all samples of i [3]. In other words, the aim of the matrix a is to ensure that the error is orthogonal to the received signal, giving rise to the lowest expected error.

Using both concepts, we can use the relationship between the channel estimate from the pilot symbol and the differential decision-feedback and double differential decision-feedback algorithms to estimate the actual channel tap. Let the actual channel tap be $\tilde{h}$ and the estimated channel tap be $\hat{h}$ and $\hat{h} = \tilde{h} + \varepsilon$. Therefore, the matrix $\boldsymbol{h}_{opt}$, where $\tilde{h} = \boldsymbol{h}_{opt}\hat{h}$ that minimizes the mean square error between the estimated channel tap and the actual channel tap is obtained as follows (derived from [3]). $R_{\hat{h}}$ is the autocorrelation function of the actual channel tap and $\boldsymbol{r}_{\tilde{h}\hat{h}}$ is the cross-correlation between the estimated and actual channel tap. The complex conjugate is added and necessary for complex signals. The symbols $\mu_{\tilde{h}}$ and $\mu_{\hat{h}}$ represent the means of the actual and estimated channel tap, respectively. Finally, using equation (7), the channel tap estimate with minimum mean square error is obtained and denoted by $\boldsymbol{h}_{est}$ (see Appendix A for derivation):

$$\begin{aligned} R_{\hat{h}}\boldsymbol{h}_{opt} &= \boldsymbol{r}^{*}_{\tilde{h}\hat{h}} \\ \boldsymbol{h}_{est} &= \boldsymbol{h}_{opt}\hat{\boldsymbol{h}} + \mu_{\tilde{h}} - \boldsymbol{h}_{opt}\mu_{\hat{h}}. \end{aligned} \tag{7}$$

3. Application to Pilot Symbol-Aided Decision Feedback Demodulation

The autocorrelation function of the channel tap can be estimated from the pilot symbol. Based on the Clark-Doppler power spectrum approach, the autocorrelation of the channel tap is a Bessel function of zero order multiplied by the received power and is calculated from

$$R_{\tilde{h}}(t) = \mathrm{E}\left[\left|\tilde{h}\right|^2\right] J_0\left(2\pi f_d t\right) \tag{8}$$

where the total received estimated power $\mathrm{E}\left[\left|\tilde{h}\right|^2\right]$ is obtained from the pilot symbol's channel tap estimate.

Assuming that the error ε has a uniform distribution and is zero at the pilot symbol's channel tap, we can derive the relationship between $\tilde{h}$ and $\mathrm{E}\left[\hat{h}\right]$ as

$$\begin{aligned} &\hat{h} = \tilde{h} + \varepsilon \\ &\mathrm{E}\left[\hat{h}\right] = \mathrm{E}\left[\tilde{h}\right] + \mathrm{E}[\varepsilon] \Rightarrow \mathrm{E}\left[\hat{h}\right] = \mathrm{E}\left[\tilde{h}\right]. \end{aligned} \tag{9}$$

The cross-correlation between the estimated channel tap from the differential decision-feedback and double differential decision-feedback algorithms and the actual channel tap can be derived from Equation (8). This assumes that the error ε is independent of the channel tap. The cross-correlation between the estimated channel tap from the differential decision-feedback and double differential decision-feedback algorithms is given by

$$\begin{aligned} \underset{\tilde{h}\hat{h}}{\boldsymbol{r}^{*}}(i) &= \mathrm{E}\left\{\left(\tilde{h}(n)\hat{h}^{*}(n+i)\right)\right\} \\ &= \mathrm{E}\left\{\left(\tilde{h}(n)\left(\tilde{h}^{*}(n+i) - \varepsilon(n+i)\right)\right)\right\} \\ &= \underset{\tilde{h}}{R}(i) - \mathrm{E}\left\{\left(\tilde{h}(n+i)\varepsilon(n+i)\right)\right\} \\ &= \underset{\tilde{h}}{R}(i) - \mu_{\tilde{h}}\mu_{\varepsilon} = \mathrm{E}\left[\left|\tilde{h}\right|^{2}\right] J_{0}(2\pi f_{m} i) - \mu_{\tilde{h}}\mu_{\varepsilon}. \end{aligned} \tag{10}$$

4. Application to Differential Decision-Feedback and Double Differential Decision-Feedback

The application of the minimum mean square error onto the differential decision-feedback and double differential decision-feedback algorithms requires the autocorrelation function of the channel tap and the estimated channel tap as well as the means for both of them. The autocorrelation of the channel tap is simply the zero-order Bessel function multiplied by the received channel tap while the autocorrelation of the estimated channel tap can be obtained from the set of estimated channel tap. To ensure that the autocorrelation function is accurate, the entire frame between two pilot symbols is used to determine both the autocorrelation function and the channel tap mean.

Using the channel estimates from the differential decision-feedback or double differential decision-feedback algorithms, we can obtain the autocorrelation and mean information and the minimum mean square error algorithm can be implemented. It is also observed that since the channel tap magnitude is small and the autocorrelation is small, the minimum mean square filter tends to amplify the magnitude of the channel tap estimate as the algorithm is implemented. Therefore, since we are only interested in the phase error, the $\boldsymbol{h}_{opt}$ is normalized by the maximum magnitude of the channel taps within the entire frame.

The new estimated channel tap is then applied to the demodulation of the l^{th} symbol directly as $\tilde{h}_l$ to correct the phase error introduced due to the fading channel. Since the minimum mean square error estimate of the channel phase error is more likely to be correct, there is no further compensation required, and the phase error estimation can be used directly for the demodulation. To balance the complexity and accuracy of the minimum mean square error algorithm, the length of $\boldsymbol{h}_{opt}$ is selected to be three. In Table 1, the average minimum mean square error of channel tap magnitude and phase of 10,000 symbols at E_b/N_o = 10 dB, used in conjunction with the differential decision-feedback algorithm is shown. Since only the phase is used in the minimum mean square error algorithm, the $\boldsymbol{h}_{opt}$ length of three is ideal.

Table 1. Tabulation of Channel Tap Magnitude and Phase Error.

Filter length	**Ave. Channel Tap Phase Error**	**Ave. Channel Tap Mag. Error**
2	0.2745	21.6862
3	-0.1589	28.6622
4	-0.2005	35.4068
5	-0.2093	39.3442

To prevent an erroneous demodulation from affecting the channel estimate in both the differential decision-feedback and double differential decision-feedback algorithms, tolerances can be included to detect drastic changes in the channel estimates. Since we know that the phase errors do not vary significantly between a few symbols, these tolerances are required to make sure that the current minimum mean square error estimate does not deviate dramatically from the previous estimate. If the new estimate from the minimum mean square error algorithm is within tolerance, the received symbol can be decoded using this new channel tap estimate. Otherwise, the previous channel tap estimate from the previous symbol is used instead. In other words, if the current estimate exceeds the tolerance, the previous minimum mean square error estimate from $l\text{-}1^{\text{th}}$ symbol is used for decoding.

When the new channel tap is used for demodulation, a new channel tap estimate for the next symbol is obtained and used subsequently. This results in an implementation where the channel estimate is constantly updated at each symbol's demodulation with minimum mean square error estimation. The functional block diagram of the new algorithm is shown in Figures 6 and 7. The MMSE estimator is added after the differential decision-feedback or double-differential decision-feedback algorithm. A buffer at the beginning of the MMSE block shown in Figure 7 is used to collect all decoded symbols with the respective channel estimates between two pilot symbols. This information is collected into a single frame and sent to the MMSE estimator. The MMSE estimator uses this information to determine the channel tap estimate's autocorrelation and carry out the MMSE calculations. The output of the MMSE estimator is $\boldsymbol{h}_{opt}$. The current and the past two channel tap estimates are then multiplied with $\boldsymbol{h}_{opt}$ to obtain the new channel estimate. The tolerance comparator then checks the new estimate against the previous symbol's MMSE estimate. If the new estimate exceeds the tolerance, the previous MMSE estimate is used instead. The symbols are then compensated with the new channel tap phase estimate, and the updated symbol is sent to the detector for decision.

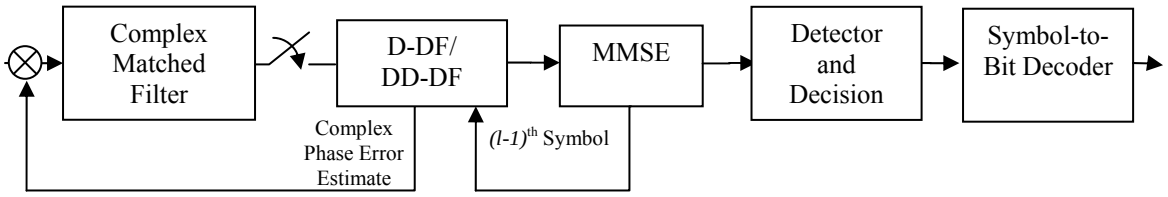


Figure 6. Functional Block Diagram of the D-DF/DD-DF with MMSE.

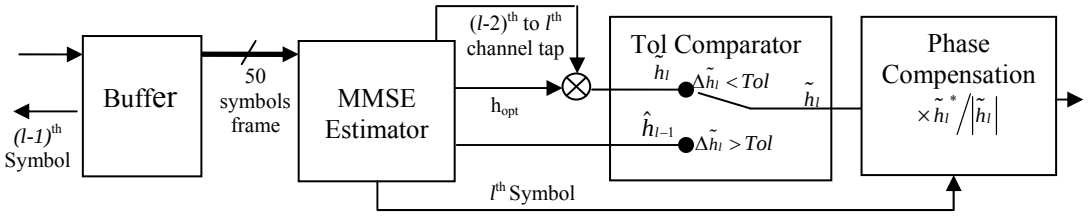


Figure 7. Functional Block Diagram of the MMSE Algorithm.

To determine the effectiveness of the new algorithm, it was implemented and evaluated with a sample data of 10,000 complex QPSK symbols, which was the sample size of a single trial. The parameters used for simulation were a symbol rate of 1,000,000 symbols per second (1 Msps), a Doppler shift of 50 Hertz and a pilot symbol interval of 50 symbols. The effectiveness of the new algorithm is shown in Figures 8 and 9. The Rayleigh channel tap generated by the simulation is shown in Figure 8. This was before additive white Gaussian noise (AWGN) is added. The channel tap estimated without and with minimum mean square error implementation is shown in Figure 9. The impact of the minimum mean square error algorithm on the channel tap phase is shown in Figure 10. The phase errors shown are between channel tap estimates of two pilot symbols. The change in the channel tap phase error estimate in Figure 10c is much less than in Figure 10b. The large transients in Figure 10(a) indicate the transition between π and $-\pi$.

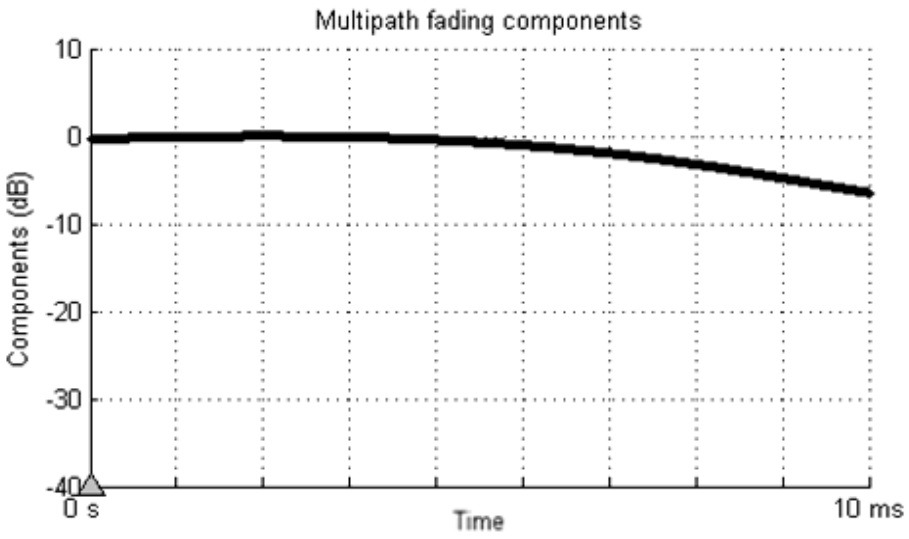


Figure 8. Channel Tap for 10,000 Symbols Generated by Simulation.

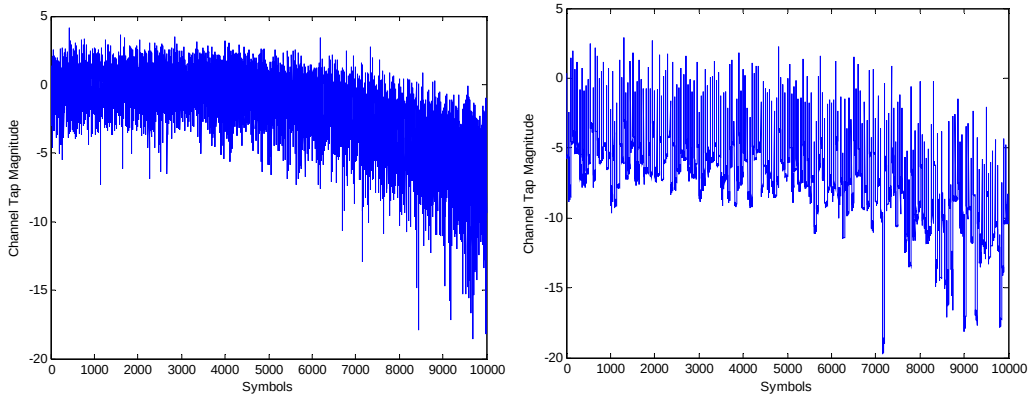
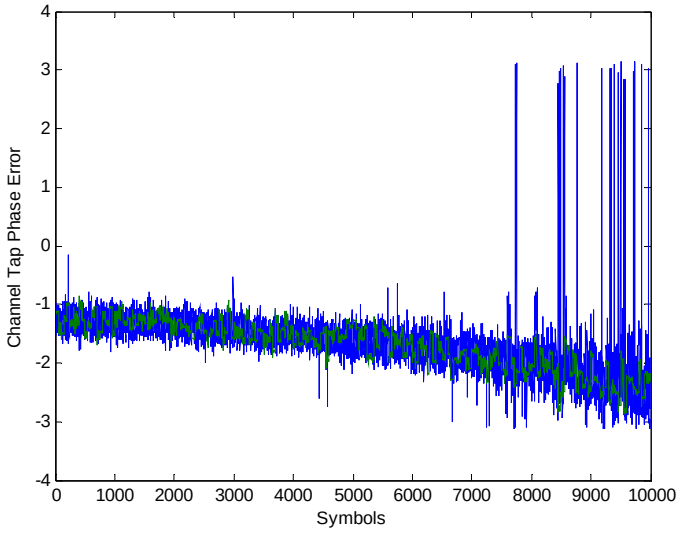
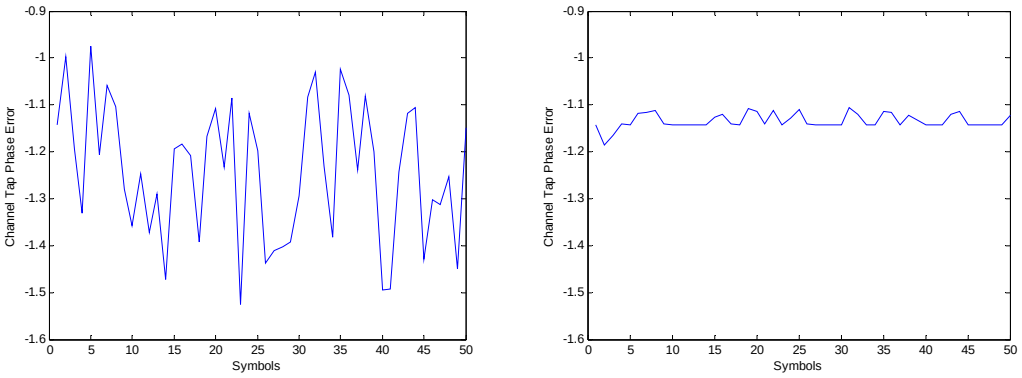


Figure 9. Channel Tap Magnitude for 10,000 Symbols Generated by D-DF without MMSE (Left) and with MMSE (Right).



(a)



(b) (c)

Figure 10. Comparison of Channel Tap Phase Errors Estimate (in Radians) (a) for 10,000 symbols, (b) without MMSE and (c) with MMSE for 50 Symbols.

B. SIMULATION RESULTS FOR CASES WITH LOW DOPPLER FREQUENCY SHIFT

1. Simulation Overview

Similar to [2], the communications toolbox in MATLAB was used to determine the performance of differential decision-feedback and double-differential decision-feedback with MMSE for single carrier and single-input single-output (SISO) systems. Adapted from the MATLAB codes provided in [2], MMSE algorithms were added to the demodulation process. Performances with and without the MMSE implemented were

derived for QPSK as a baseline and extended to 16-QAM, 64-QAM and 256-QAM. The latter modulations were selected based on their relevance to fourth generation mobile communications systems such as Long Term Evolution (LTE) and WiMAX.

The parameters used for simulations were selected to give the best performance when comparing D-DF and DD-DF with and without MMSE. The symbol rate, Doppler shift and pilot symbol interval were selected to be within the coherence bandwidth and time, giving rise to slow frequency non-selective (flat) fading. The parameters used were a symbol rate of 1,000,000 symbols per second (1 Msps), a Doppler shift of 50 Hz and a pilot symbol interval of 50 symbols. A minimum of 2,000 trials, each with 10,000 symbols, was used to determine the bit error rate for each E_b/N_o. The E_b/N_o ranges from 10 to 60 dB.

2. QPSK

The differential decision-feedback algorithm with MMSE introduced showed insignificant improvement to the bit error rate (BER) performances. This could be due to the fact that the differential decision-feedback algorithm is already very effective in ensuring that the phase correction is sufficient to rotate the received symbol into the correct decision region. In this case, the MMSE was unable to enhance the estimation further. The BER plot of differential decision-feedback with and without MMSE implemented is shown in Figure 11. The conclusion is the same for double differential decision-feedback as shown in Figure 12.

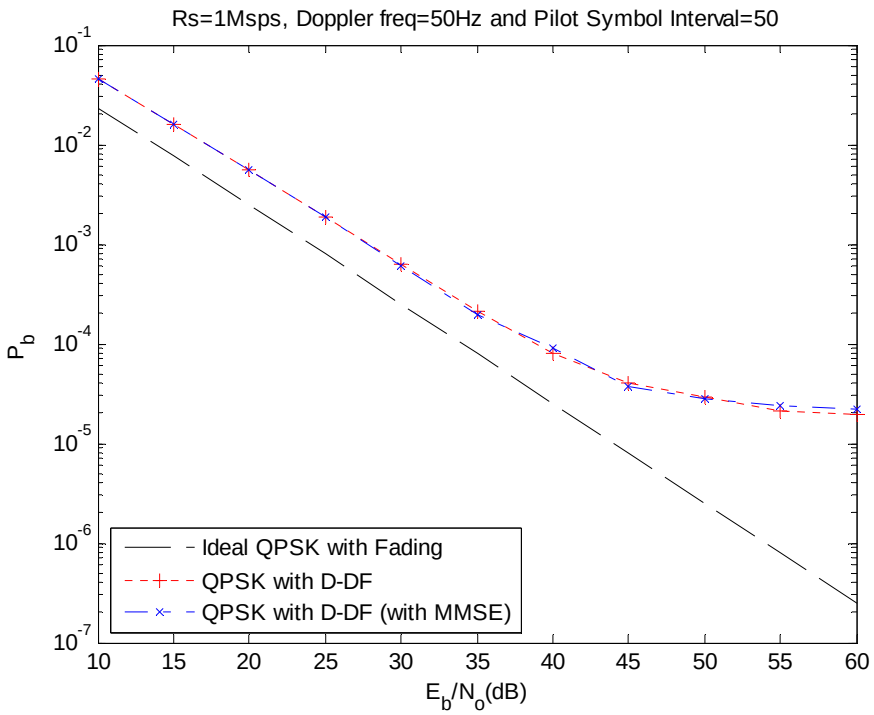


Figure 11. BER of QPSK with D-DF Algorithm with and without MMSE Filter.

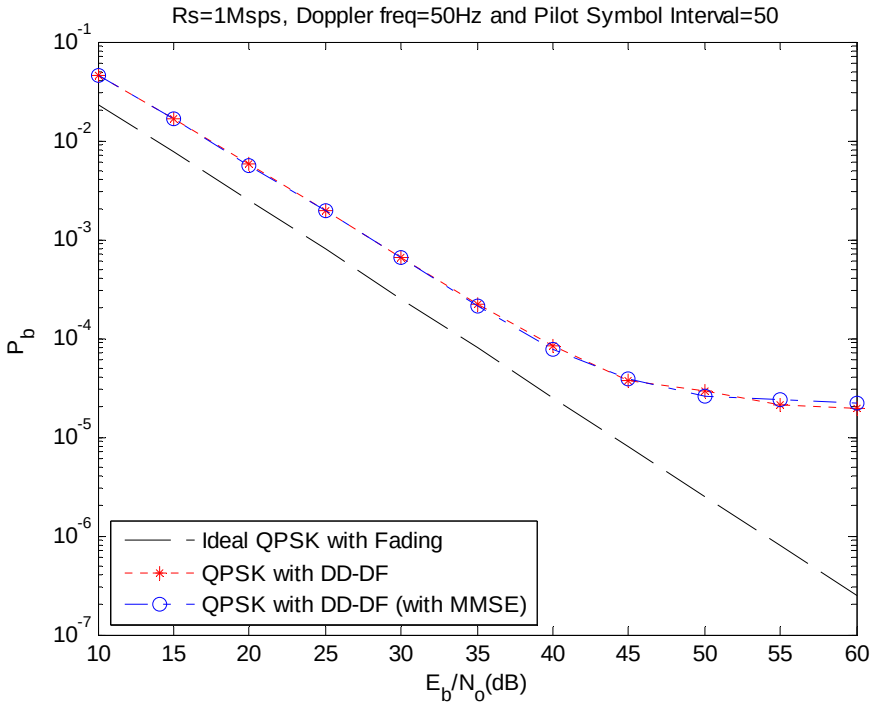


Figure 12. BER of DD-DF Algorithm with and without MMSE Filter.

3. QAM

Extending the studies to QAM, we implemented the minimum mean square error algorithm for 16-QAM, 64-QAM and 256-QAM with differential decision-feedback and double differential decision-feedback algorithms. The 64-QAM and 256-QAM were studied since they are used in the newer communications standards such as WiMax and LTE. The performances were compared, and it was shown that with higher order modulations, which are most susceptible to phase errors, the minimum mean square error filter was proven to be effective.

a. 16-QAM

When minimum mean square error is applied to the 16-QAM modulation, there is approximately 0.5 dB improvement to the performance obtained through the differential decision-feedback and 1 dB improvement with the double differential decision-feedback algorithm as shown in Figures 13 and 14, respectively. One common characteristic of the new bit error performances with the new minimum mean square error algorithm is that the performance with higher signal-to-noise ratios, generally after 35 dB, is worse than the performance without the minimum mean square error implementation. This could be due to the fact that at larger E_b/N_o, the channel estimate from the differential decision-feedback and double differential decision-feedback algorithms is relatively noise-free, and the minimum mean square error algorithm may have overcompensated for the phase errors. In other words, at larger E_b/N_o, the variance of the channel estimate reduces dramatically and, therefore, the tolerances to detect surges in channel estimate need to be reduced accordingly. Comparing both differential decision-feedback and double differential decision-feedback algorithms with minimum mean square error estimation, we see that the bit error rate performance of the double differential decision-feedback algorithm is worse than that of the differential decision-feedback algorithm as shown in Figure 15.

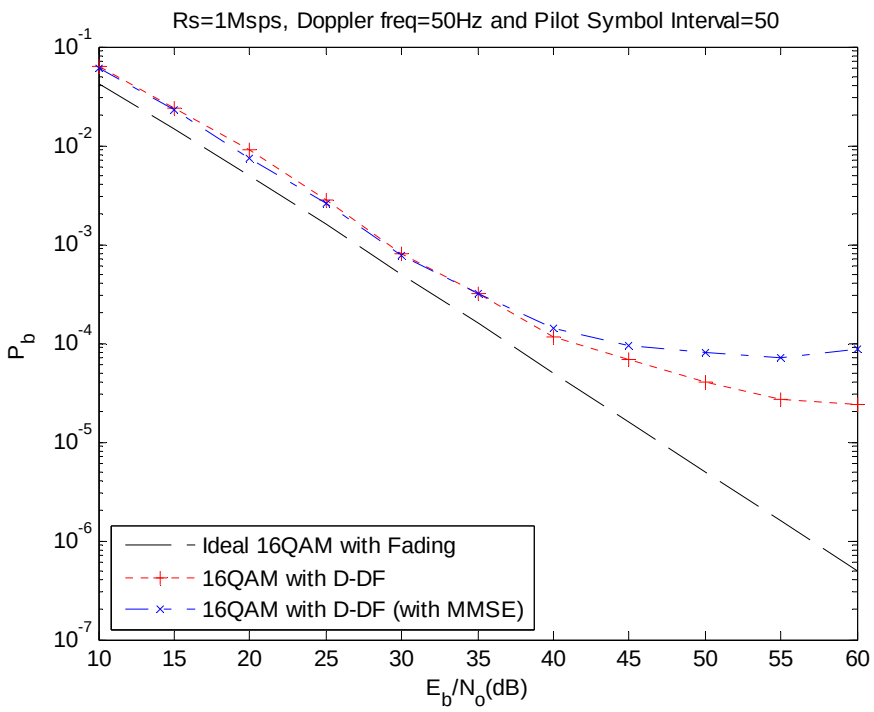


Figure 13. BER of 16-QAM with D-DF Algorithm with and without MMSE Filter.

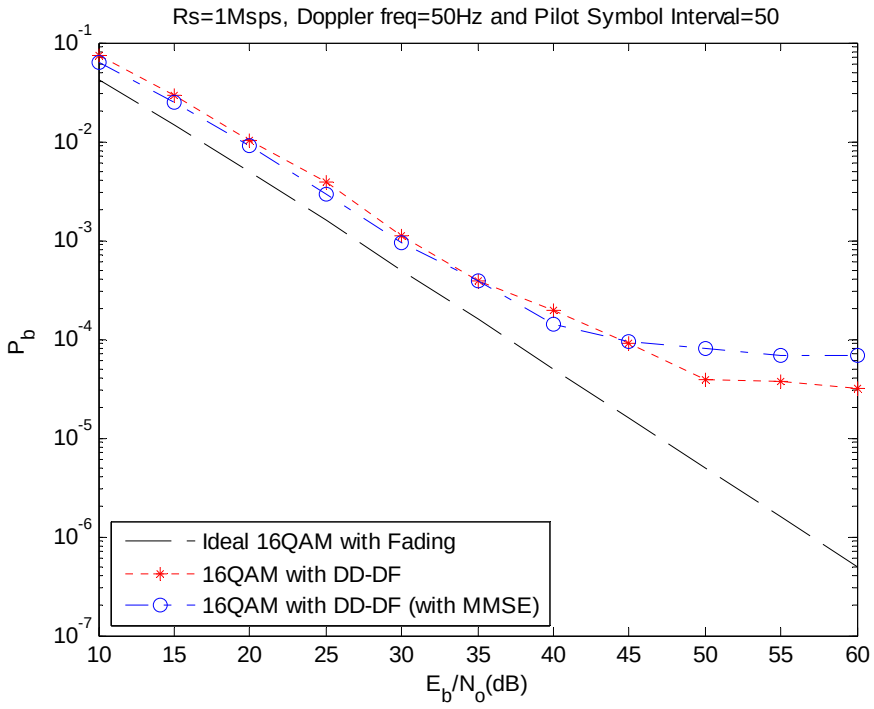


Figure 14. BER of 16-QAM with DD-DF Algorithm with and without MMSE Filter.

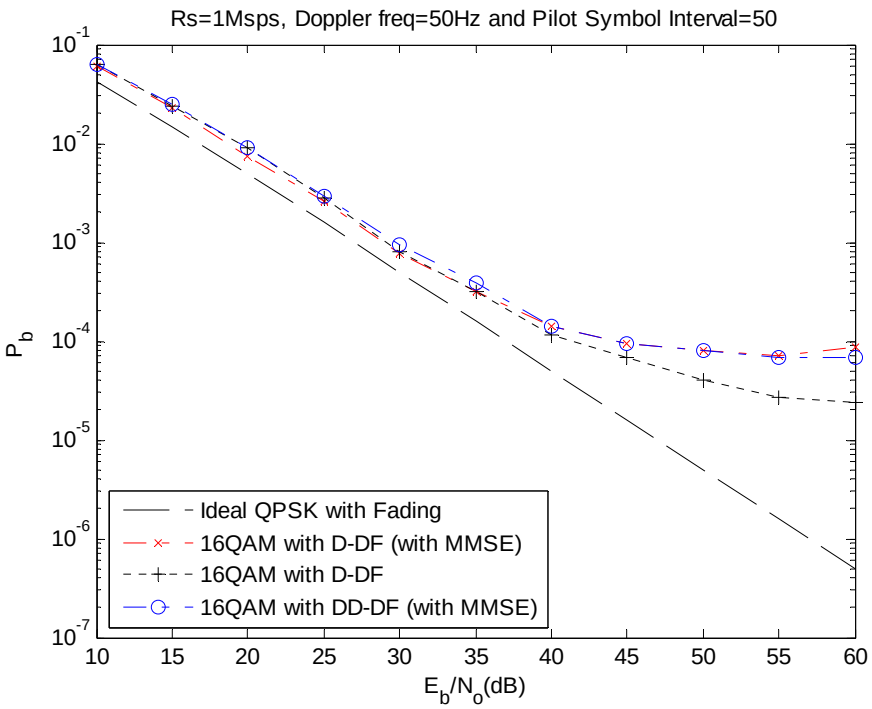


Figure 15. Comparison of 16-QAM with D-DF and DD-DF Algorithms both with MMSE Filter.

b. 64-QAM

When the minimum mean square error algorithm is applied to 64-QAM with differential decision-feedback and double differential decision-feedback, there is a greater improvement to the bit error rates. It can be observed that there is a 2-3 dB improvement in bit error rates at the lower region of E_b/N_o. The performances of 64-QAM with differential decision-feedback and double differential decision-feedback with minimum mean square error implementation are shown in Figures 16 and 17. Also, the bit error rates at the higher region of E_b/N_o are slightly higher than that of the differential decision-feedback algorithm. The other observation is that the bit error rates of double differential decision-feedback with minimum mean square error at higher E_b/N_o are decreasing and approaching the performances without minimum mean square error. The comparison of both differential decision-feedback and double differential decision-feedback with minimum mean square error is shown in Figure 18.

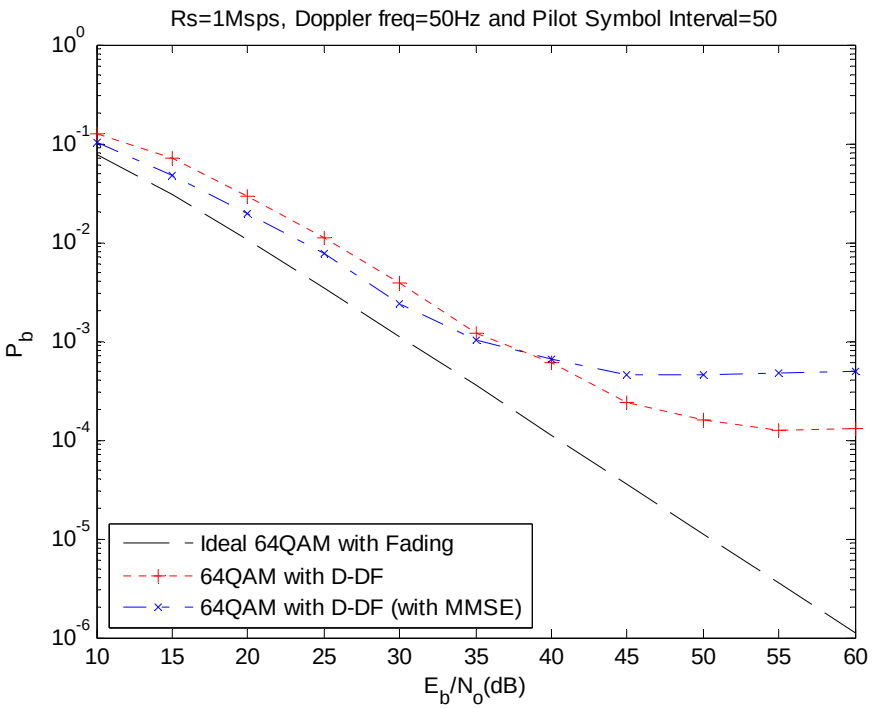


Figure 16. BER of 64-QAM with D-DF Algorithm with and without MMSE Filter.

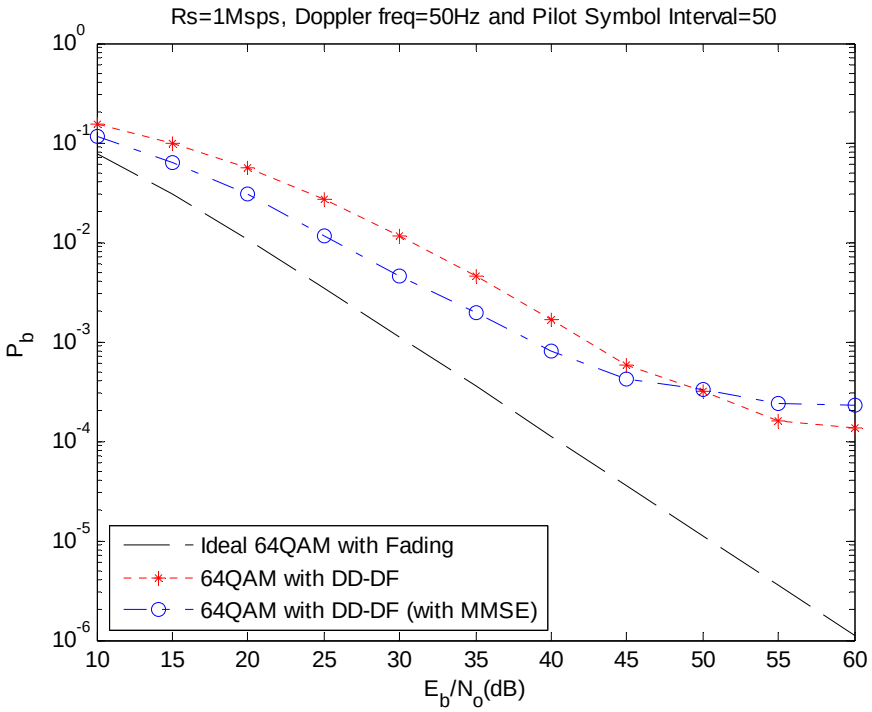


Figure 17. BER of 64-QAM with DD-DF Algorithm with and without MMSE Filter.

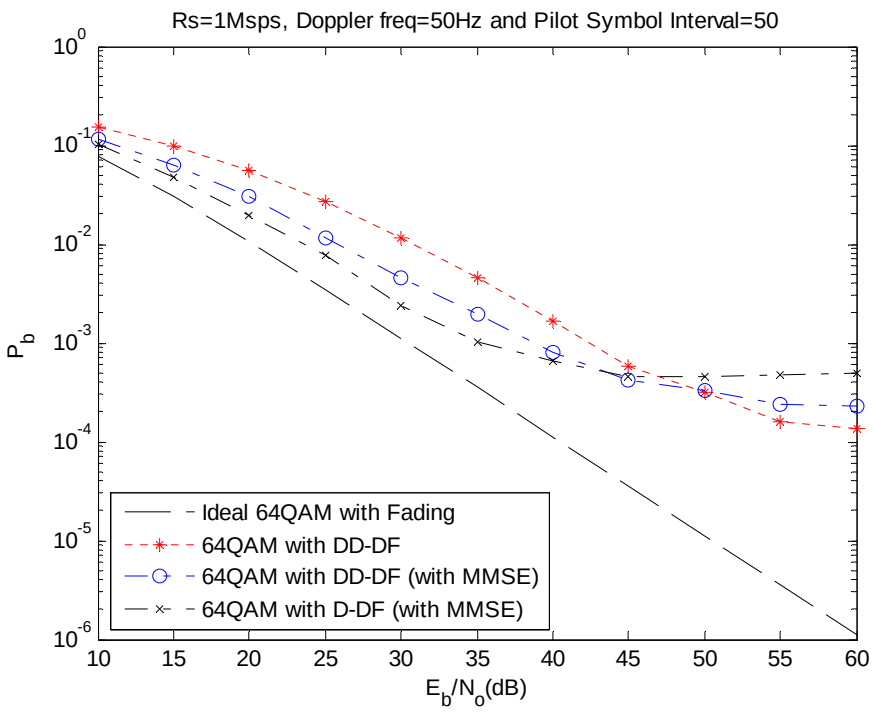


Figure 18. Comparison of 64-QAM with D-DF and DD-DF Algorithms both with MMSE Filter.

c. 256-QAM

As for 256-QAM, the minimum mean square error algorithm is highly effective for the double differential decision-feedback algorithm. Similar to previous cases, when minimum mean square error is implemented for the differential decision-feedback algorithm, the bit error rates at higher E_b/N_o are worse, as shown in Figure 19. In fact, as the order of modulation increases, the degradation in performances at higher E_b/N_o is worse. On the other hand, for double differential decision-feedback with minimum mean square error, the performance degradation at higher E_b/N_o caused by the new algorithm reduces with higher order of modulation. Even so, there is a slight compromise in performance at the lower E_b/N_o. The bit error performance of double differential decision-feedback with minimum mean square error is shown in Figure 20. In fact, the minimum mean square error algorithm with double differential decision-feedback was so effective in improving the bit error rates that the improvement could be

up to 15 dB as compared to the demodulation without minimum mean square error. When compared to the bit error rate performance of the differential decision-feedback algorithm, there is only approximately 5 dB improvement, as shown in Figure 21.

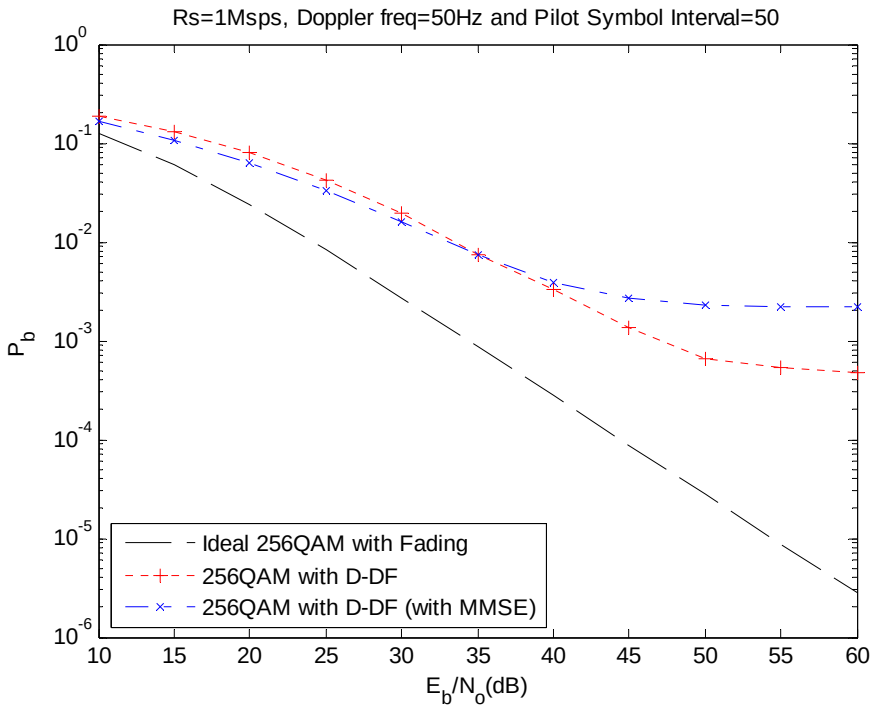


Figure 19. BER of 256-QAM with D-DF Algorithm with and without MMSE Filter.

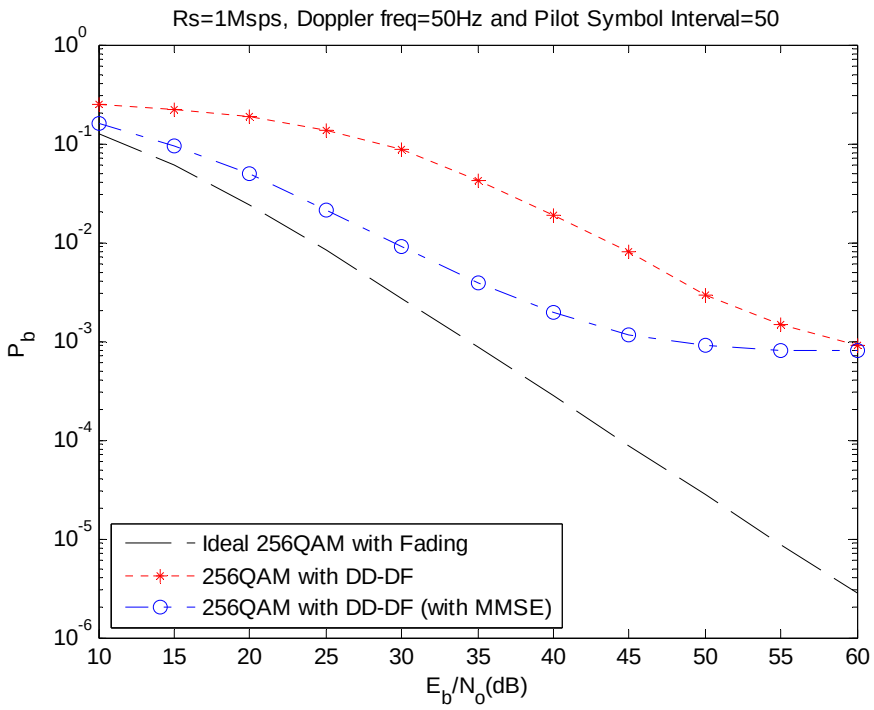


Figure 20. BER of 256-QAM with DD-DF Algorithm with and without MMSE Filter.

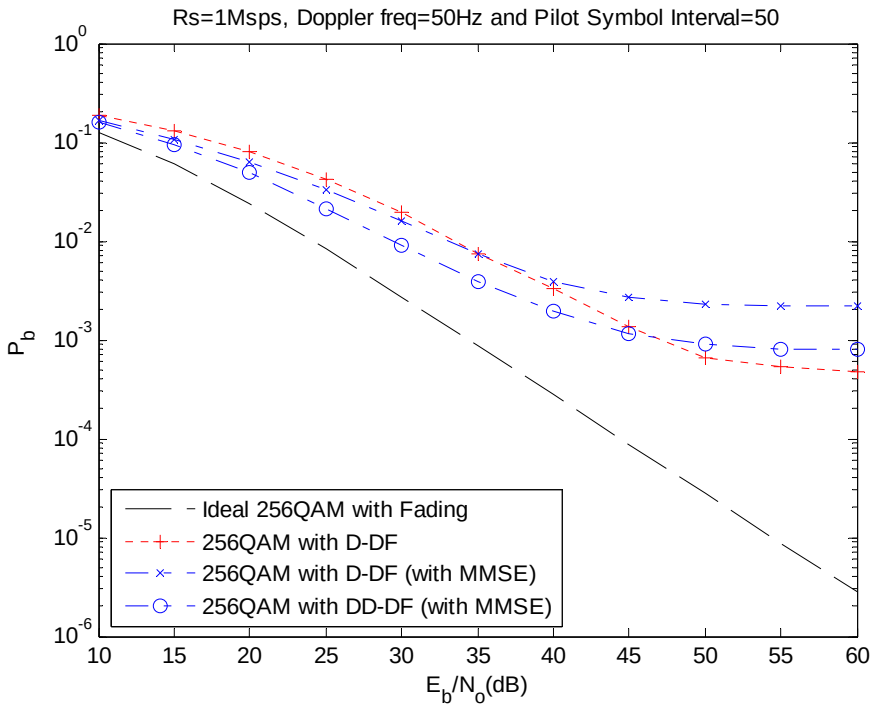


Figure 21. Comparison of 256-QAM with D-DF and DD-DF Algorithms both with MMSE Filter.

d. 512-QAM and 1024-QAM

Similar to the case at 256-QAM, the bit error rate performances for double differential decision-feedback with the minimum mean square error algorithm performed the best as compared to the other algorithms at lower E_b/N_o as shown in Figures 22 and 23. The bit error rates also tapered off at a higher level for larger E_b/N_o. The parameters used for the simulations in Figures 22 and 23 were a symbol rate of 1,000,000 symbols per second (1 Msps), a Doppler shift of 50 Hertz, and a pilot symbol interval of 50 symbols.

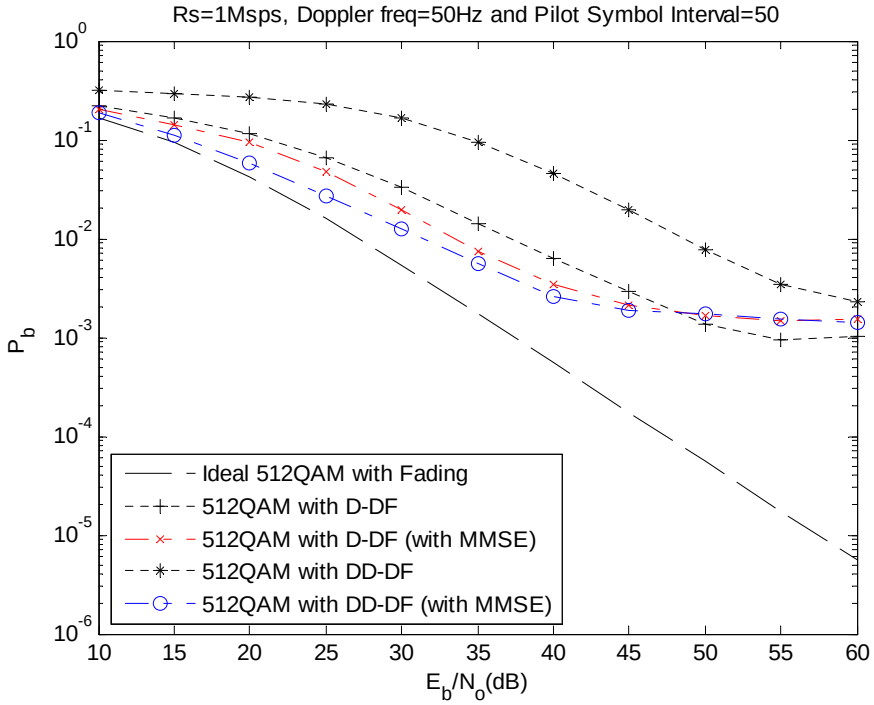


Figure 22. BER of 512-QAM with D-DF and DD-DF Algorithms with and without MMSE Filter.

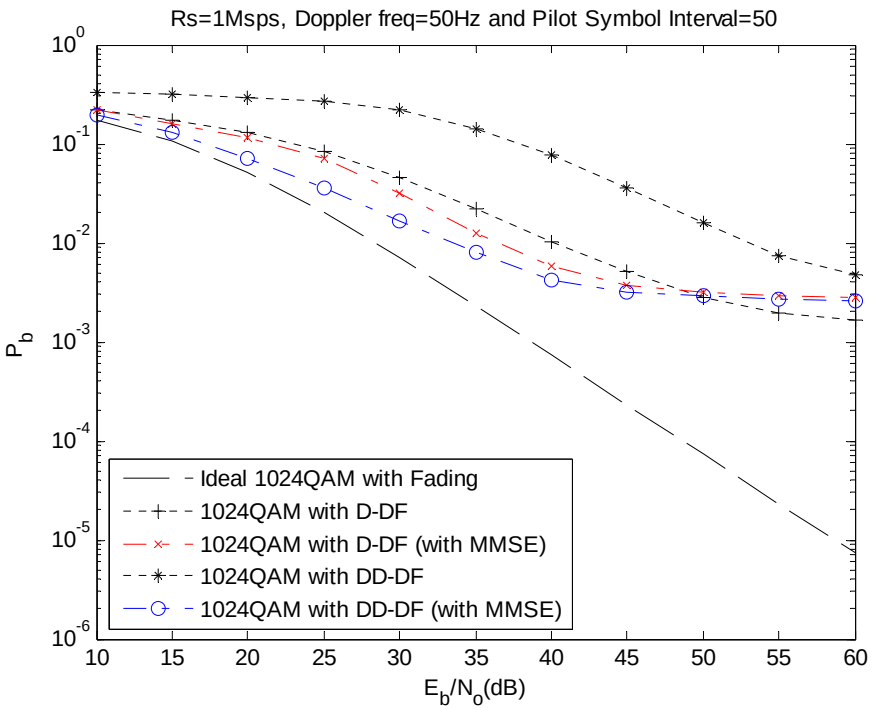


Figure 23. BER of 1024-QAM with D-DF and DD-DF Algorithms with and without MMSE Filter.

4. Comparison of the Minimum Mean Square Error Implementation on Differential Decision-Feedback and Double-Differential Decision-Feedback Algorithms

Comparing the performances of differential decision-feedback and double-differential decision-feedback with and without minimum mean square error, we conclude that the minimum mean square error algorithm is effective for all modulations at lower E_b/N_o. However, the effectiveness degrades with increasing E_b/N_o. The effectiveness of the minimum mean square error algorithm for different E_b/N_o and modulation schemes with differential decision-feedback is shown in Table 2. The effectiveness of the minimum mean square error algorithm for the double-differential decision-feedback algorithm is shown in Table 3. The most effective algorithms for the different modulation schemes are shown in Table 4.

Table 2. Tabulation of MMSE Effectiveness on D-DF (Y= Effective; N= Not Effective).

SNR (dB)	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**
QPSK	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N
16-QAM	Y	Y	Y	Y	Y	Y	N	N	N	N	N
64-QAM	Y	Y	Y	Y	Y	Y	N	N	N	N	N
256-QAM	Y	Y	Y	Y	Y	Y	N	N	N	N	N
512-QAM	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N
1024-QAM	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N

Table 3. Tabulation of MMSE Effectiveness on DD-DF (Y= Effective; N= Not Effective).

SNR (dB)	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**
QPSK	Y	Y	Y	Y	Y	N	N	N	N	N	N
16-QAM	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N
64-QAM	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N
256-QAM	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
512-QAM	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
1024-QAM	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Table 4. Tabulation of Most Effective Algorithms for Different Modulations.

Modulation	Most Effective Algorithm (Low SNR)	Most Effective Algorithm (High SNR)
QPSK	D-DF with MMSE	D-DF
16-QAM	D-DF with MMSE	D-DF
64-QAM	D-DF with MMSE	D-DF
256-QAM	DD-DF with MMSE	D-DF
512-QAM	DD-DF with MMSE	D-DF
1024-QAM	DD-DF with MMSE	D-DF

C. SIMULATION RESULTS FOR CASES WITH HIGH DOPPLER FREQUENCY SHIFTS

1. Simulation Overview

Using the same simulation code and packages as in the previous section, we repeated the simulations for QPSK and 64-QAM were repeated but with high Doppler shifts. This results in a channel that is rapidly changing and, depending on the channel estimation time used, the channel can be fast fading between two pilot symbols. Two different Doppler frequency shifts (100 Hz and 200 Hz) were tested to fully understand the performances of the algorithms. For both cases of Doppler frequency shifts, the symbol rate was reduced to 10,000 symbols per second (10 ksps). This results in a longer time period between two pilot symbols, causing it to be closer to the coherence time. This causes the channel to approach a fast fading channel. The coherence times for Doppler shift of 100 Hz and 200 Hz are 12.5 and 25 symbols, respectively. Therefore, for both cases of Doppler frequency shifts, the pilot symbol intervals of 6, 12, 25 and 50 symbols were used to test the algorithms.

Due to the high bit error rates, the number of trials and the sample size per trial were reduced to 2,000 trials and 1,000 symbols per trial. This resulted in a total of 2,000,000 symbols, which provided sufficiently accurate bit error rates. The bit error rates presented had very slight differences, and to clearly distinguish the performance differences, the bit error rate plots were magnified in the y-axis to distinguish the performances of different algorithms.

Since the correlation between the channel taps of two symbols in a fast fading channel is low and MMSE is computed based on correlation of past samples, MMSE is expected to provide poor performance and create more errors than correct them. Therefore, the only means to combat fast fading channels is to reduce throughput and transmit pilot symbols more frequently or to use dedicated pilot channels in the case of orthogonal frequency division multiplexing (OFDM).

2. QPSK with High Doppler Frequency Shifts of 100 Hz

For QPSK, the performance of the double-differential decision-feedback algorithm is better than that of the differential decision-feedback algorithm, as shown in Figures 24 to 27. At a pilot symbol interval of six, the bit error rate performances for all the algorithms are similar. For Doppler frequency shifts of 100 Hz, the minimum mean square error algorithm is not effective for all pilot symbol intervals of 12 and above. In such extreme cases where the fading channel is much faster than the channel estimation rate, the minimum mean square filtering seemed to create more errors than correction. In addition, as the pilot symbol interval increased, the minimum mean square error implementation resulted in poorer performance, as shown in Figure 26. Also, at a pilot symbol interval of six as shown in Figure 24, the bit error rate performance is much better. This is expected since throughput is sacrificed for better bit error rate performance.

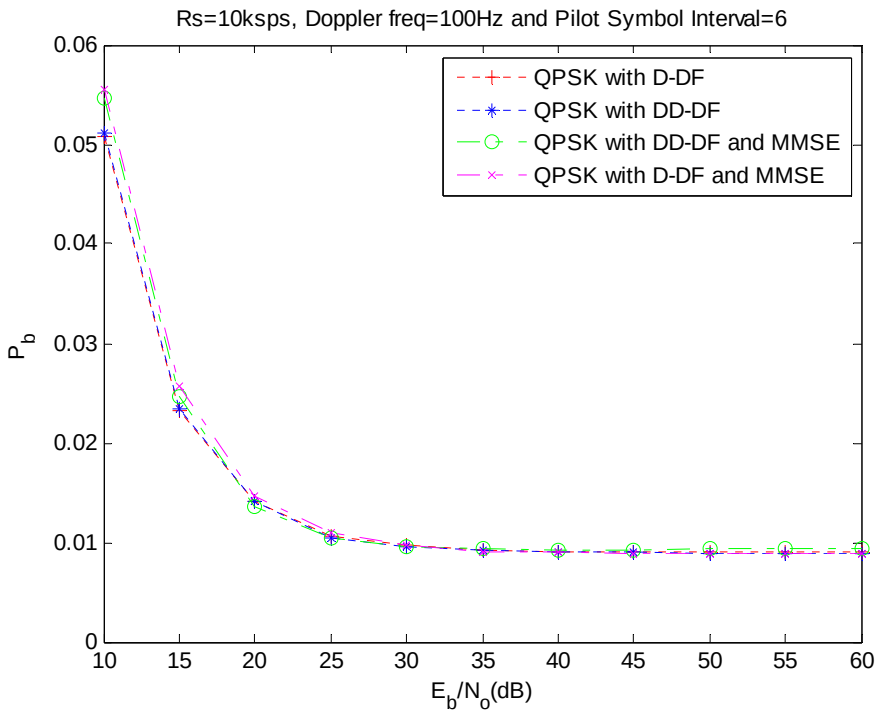


Figure 24. BER of QPSK with 100 Hz Doppler Shift and Pilot Symbol Interval of six.

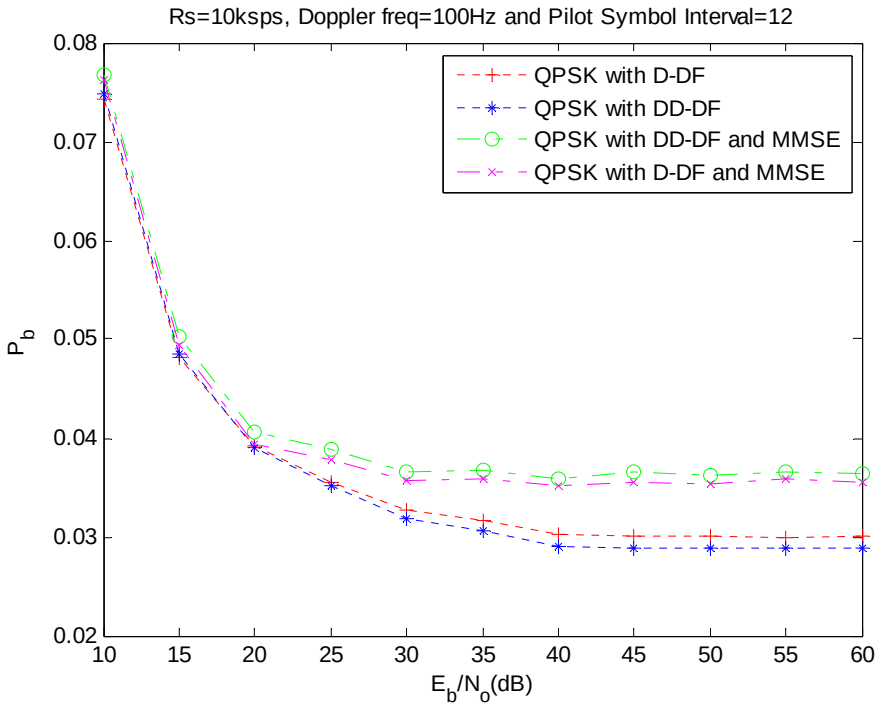


Figure 25. BER of QPSK with 100 Hz Doppler Shift and Pilot Symbol Interval of 12.

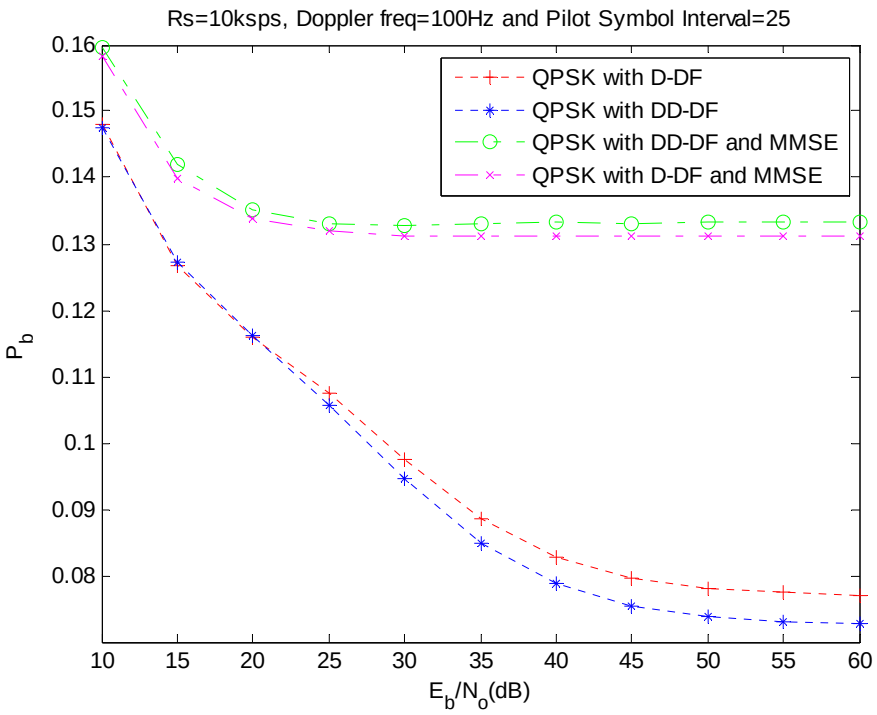


Figure 26. BER of QPSK with 100 Hz Doppler Shift and Pilot Symbol Interval of 25.

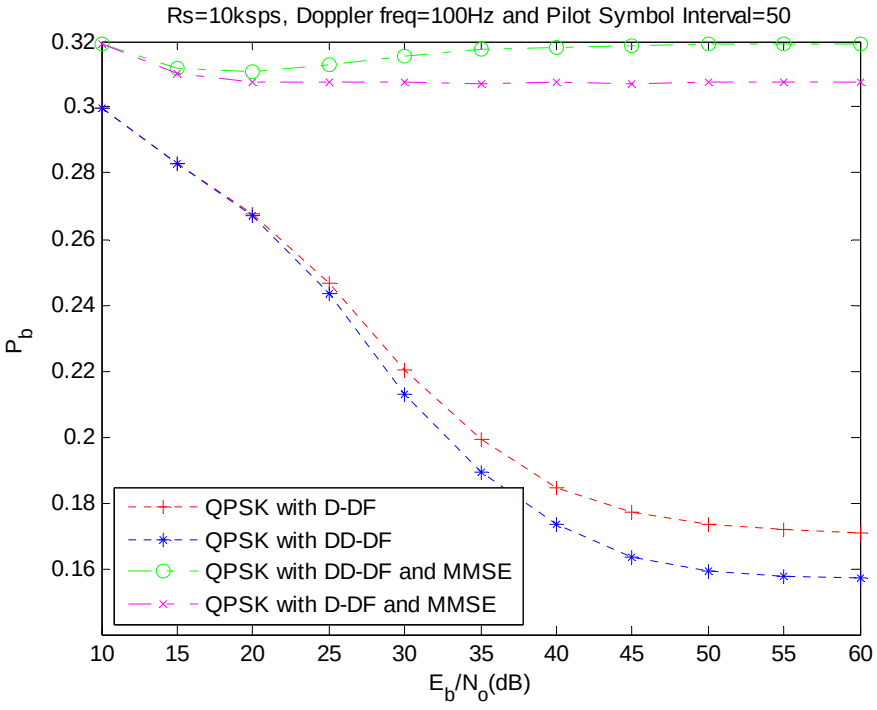


Figure 27. BER of QPSK with 100 Hz Doppler Shift and Pilot Symbol Interval of 50.

3. QPSK with High Doppler Frequency Shifts of 200 Hz

Similar to the previous scenario, the performance of the double-differential decision-feedback algorithm is better than that of the differential decision-feedback algorithm, and the minimum mean square error algorithm is not effective especially for pilot symbol intervals above six symbols, as can be seen in Figures 28 to 31. However, generally, the difference between the bit error rate performance of the double-differential decision-feedback algorithm and the differential decision-feedback algorithm is higher for lower Doppler frequency shifts. The maximum improvement of the double-differential decision-feedback algorithm for a Doppler shift of 100 Hz is approximately 10% better than that of the differential decision-feedback algorithm. At 200 Hz, the maximum is only 5%.

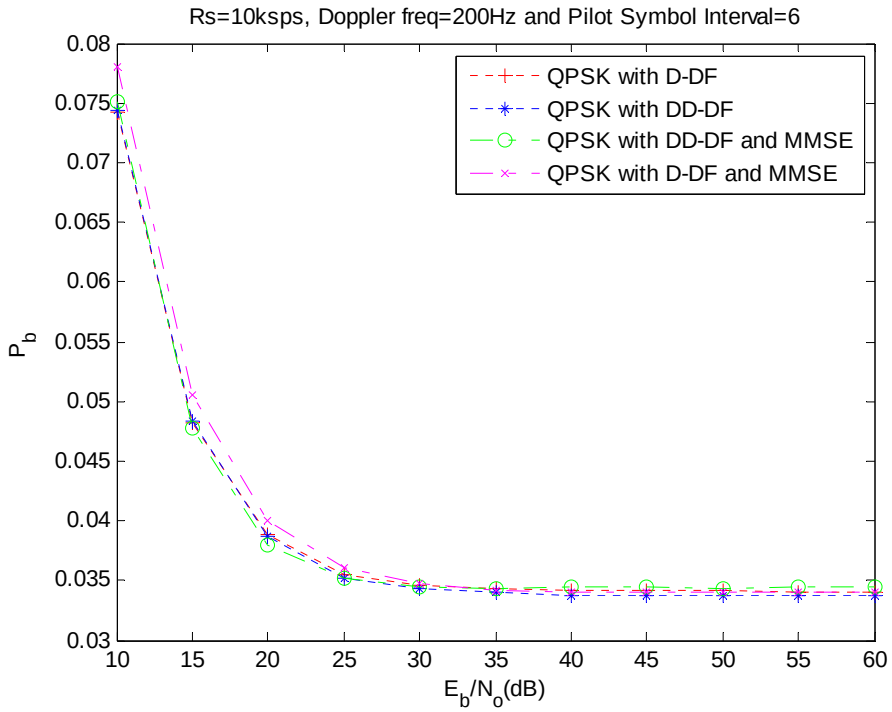


Figure 28. BER of QPSK with 200 Hz Doppler Shift and Pilot Symbol Interval of six.

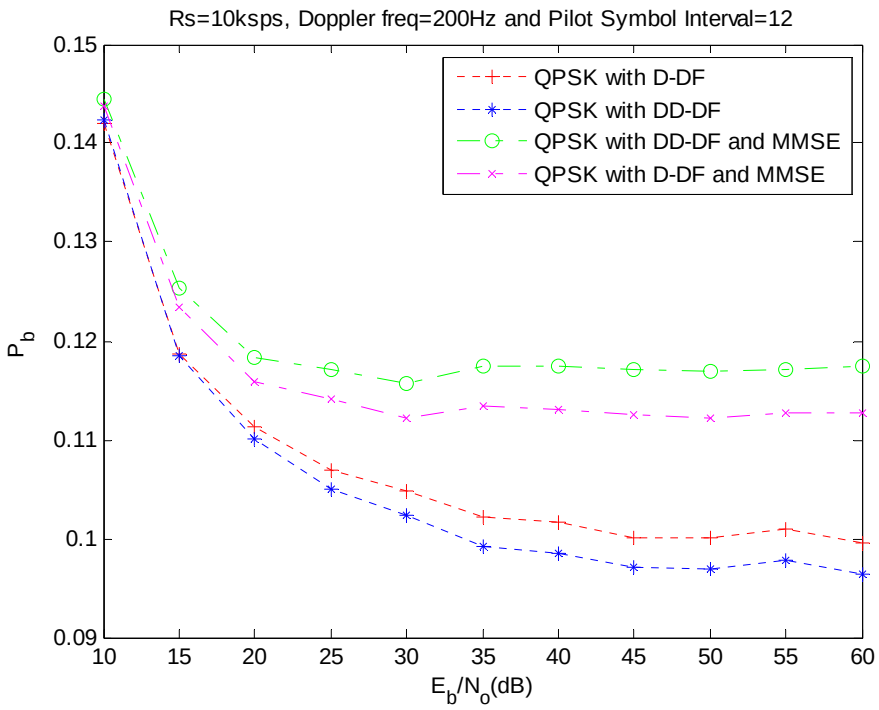


Figure 29. BER of QPSK with 200 Hz Doppler Shift and Pilot Symbol Interval of 12.

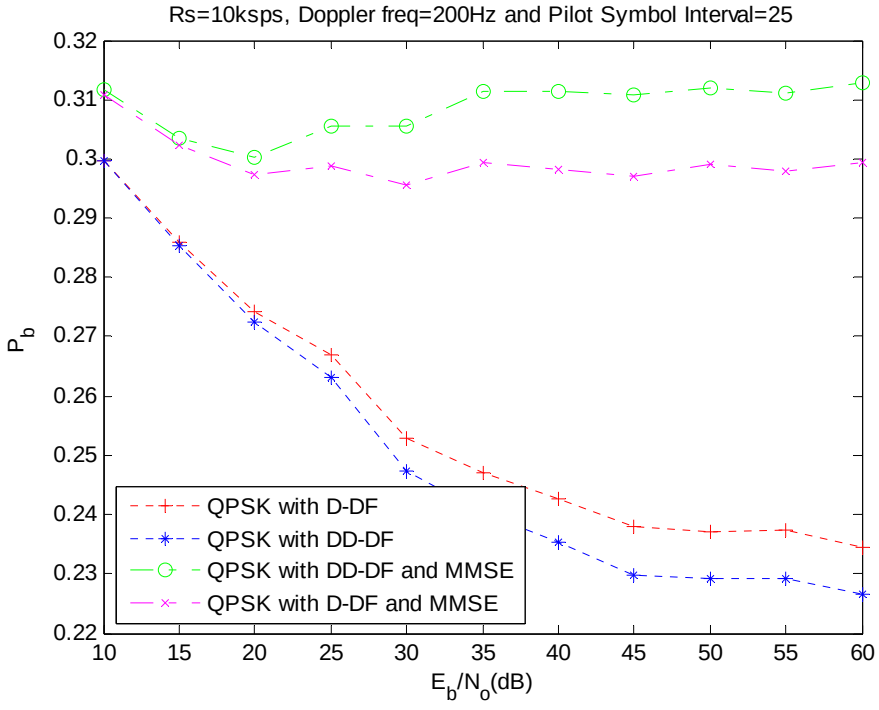


Figure 30. BER of QPSK with 200 Hz Doppler Shift and Pilot Symbol Interval of 25.

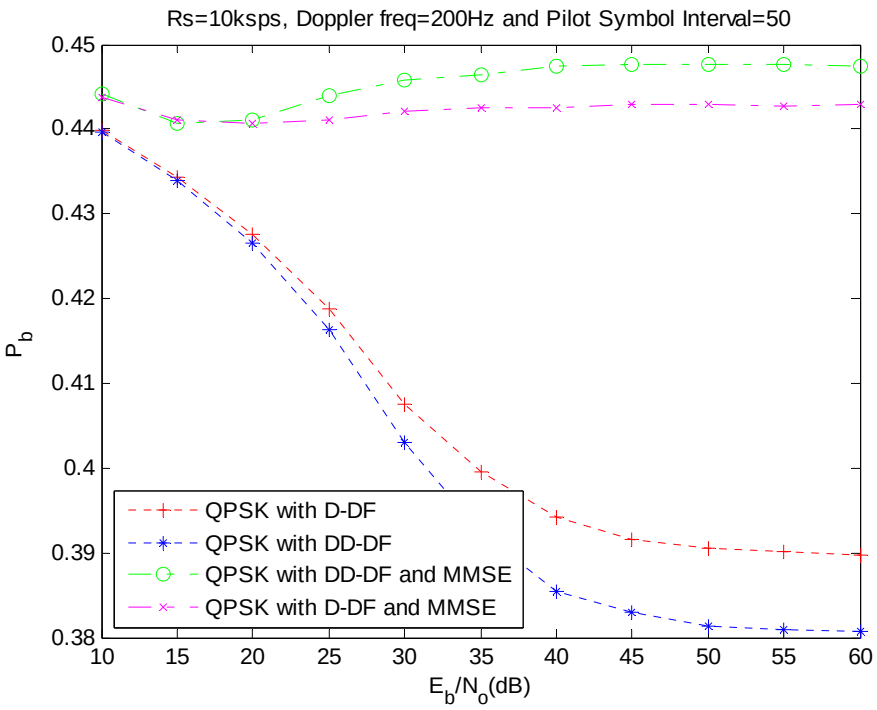


Figure 31. BER of QPSK with 200 Hz Doppler Shift and Pilot Symbol Interval of 50.

4. 64-QAM with High Doppler Frequency Shifts of 100 Hz

As for 64-QAM, the bit error rate performances of the double-differential decision-feedback algorithm are most effective at higher E_b/N_o. At lower E_b/N_o, the differential decision-feedback algorithm performed slightly better than the double-differential decision-feedback algorithm. The minimum mean square error algorithm for both differential decision-feedback and double-differential decision-feedback algorithms was not effective in improving the bit error rate, similar to the QPSK case with long pilot symbol intervals. However, for 64-QAM, the double-differential decision-feedback with minimum mean square error algorithm performed better than differential decision-feedback with the minimum mean square error algorithm. This is consistent with the findings when lower Doppler shifts were used. Double-differential decision-feedback with the minimum mean square error algorithm generally performed better at higher order modulation schemes. The bit error rate performances for 64-QAM for the four different pilot symbol intervals are shown in Figures 32 to 35.

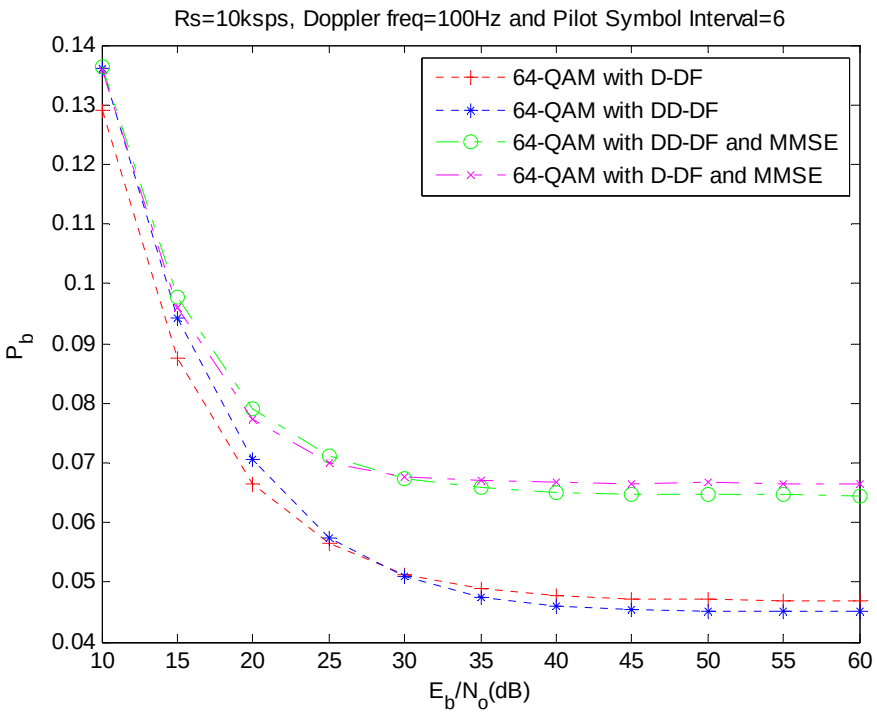


Figure 32. BER of 64-QAM with 100 Hz Doppler Shift and Pilot Symbol Interval of 12.

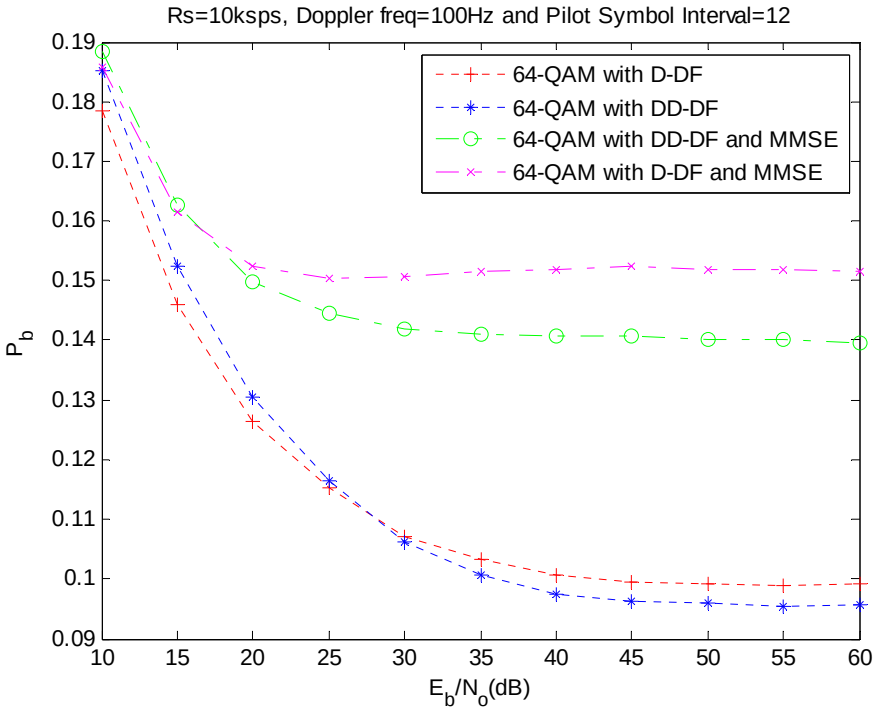


Figure 33. BER of 64-QAM with 100 Hz Doppler Shift and Pilot Symbol Interval of 12.

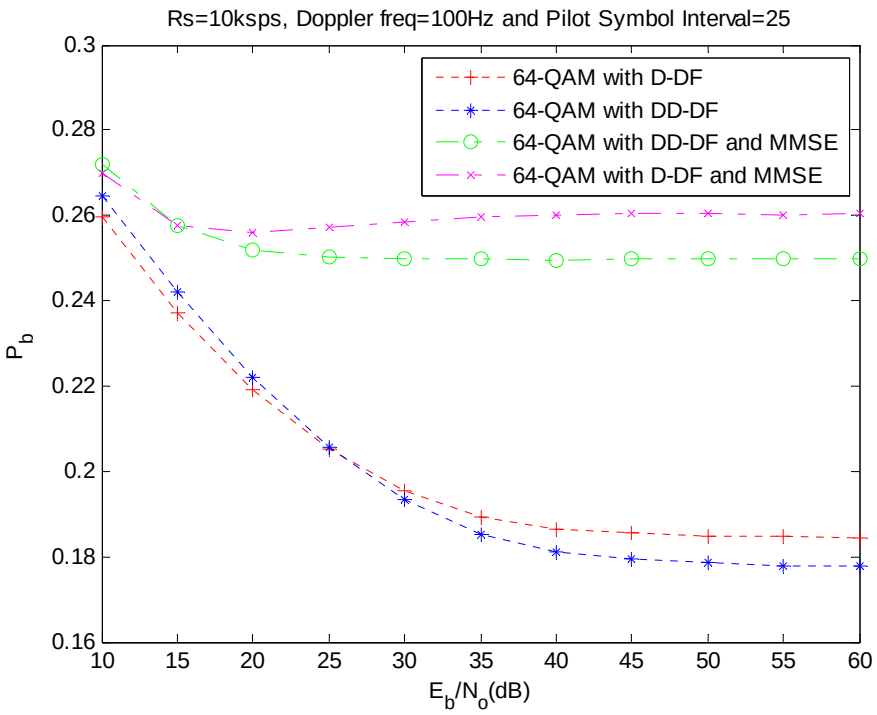


Figure 34. BER of 64-QAM with 100 Hz Doppler Shift and Pilot Symbol Interval of 25.

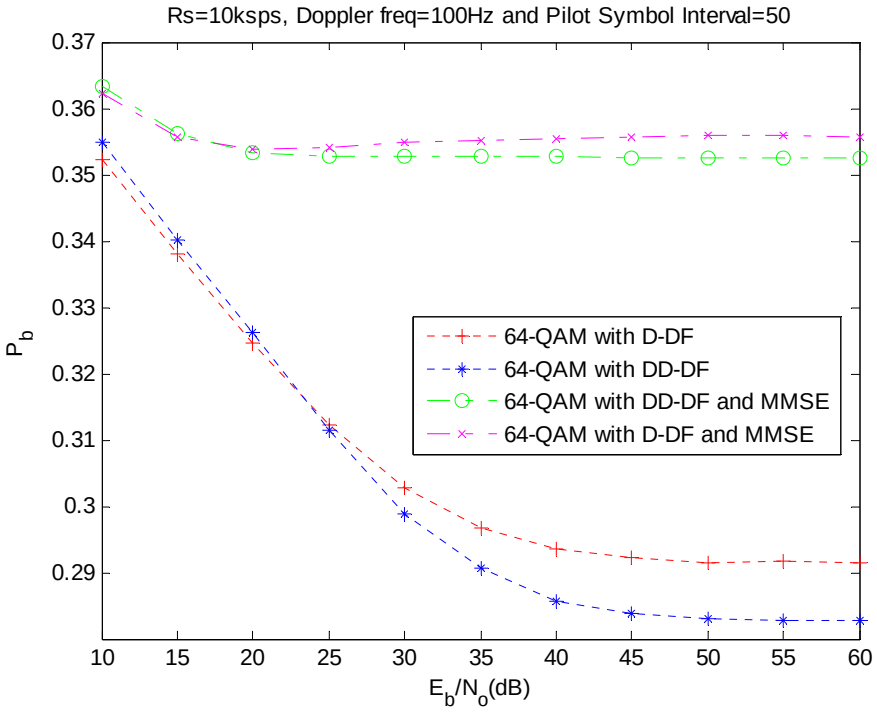


Figure 35. BER of 64-QAM with 100 Hz Doppler Shift and Pilot Symbol Interval of 50.

5. 64-QAM with High Doppler Frequency Shifts of 200 Hz

The conclusion for 64-QAM at a Doppler shift of 200 Hz is similar to that with the Doppler shift of 100 Hz. However, the difference between the two differential decision-feedback algorithms at lower E_b/N_o decreases with an increasing pilot symbol interval. In the case where the pilot symbol interval is 50, the bit error rate performance difference is almost negligible. The bit error rate performances for 64-QAM with a Doppler shift of 200 Hz for the four different pilot symbol intervals are shown in Figures 36 to 39.

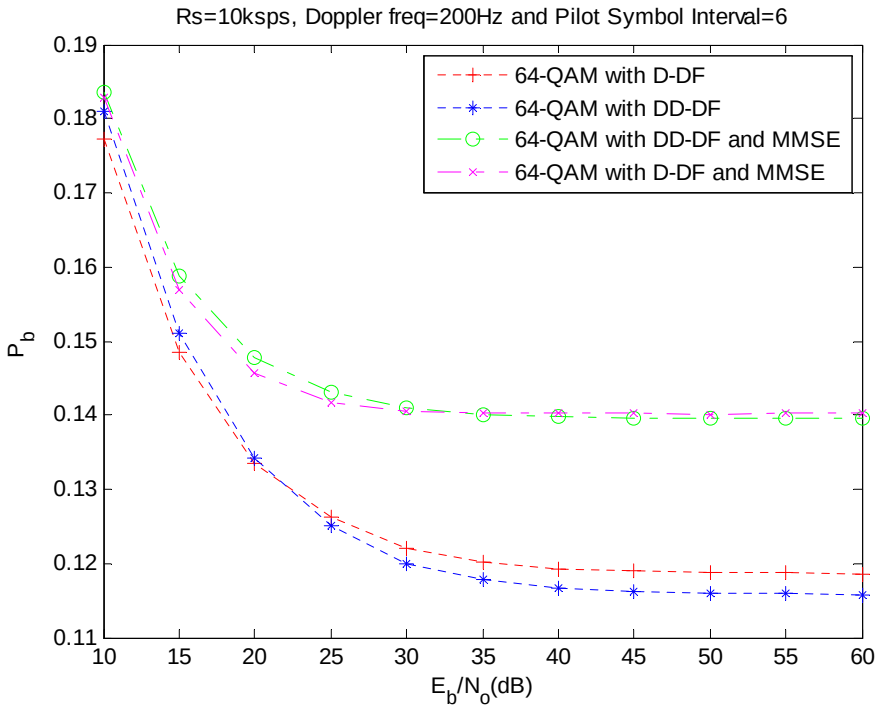


Figure 36. BER of 64-QAM with 200 Hz Doppler Shift and Pilot Symbol Interval of six.

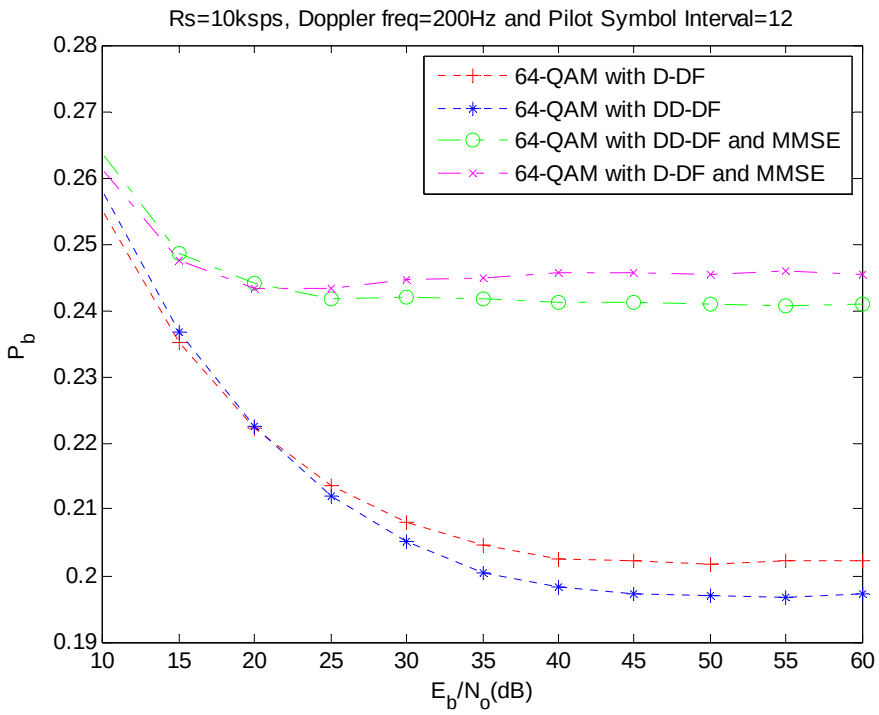


Figure 37. BER of 64-QAM with 200 Hz Doppler Shift and Pilot Symbol Interval of 12.

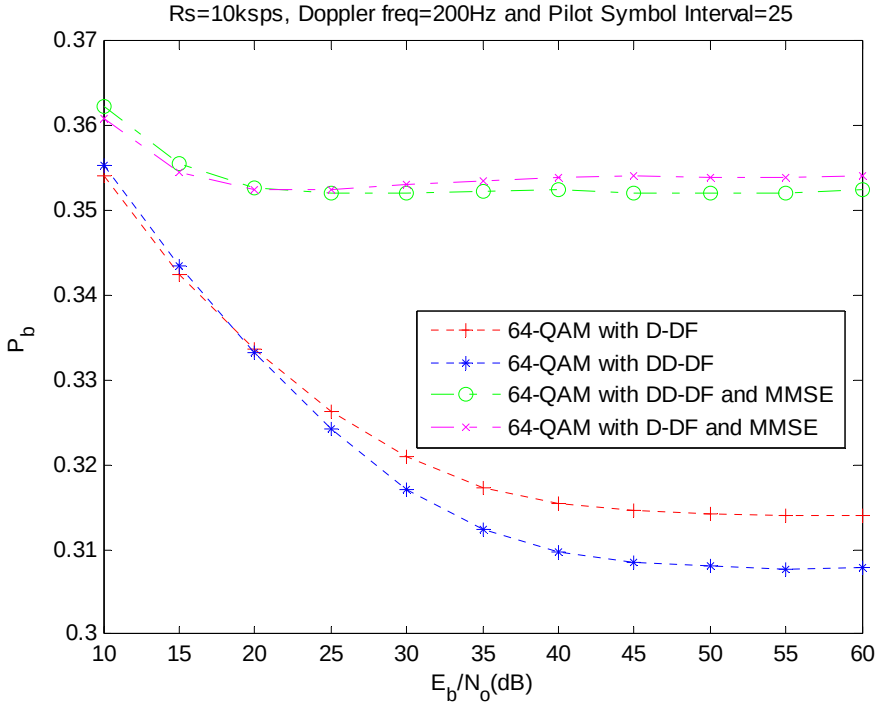


Figure 38. BER of 64-QAM with 200 Hz Doppler Shift and Pilot Symbol Interval of 25.

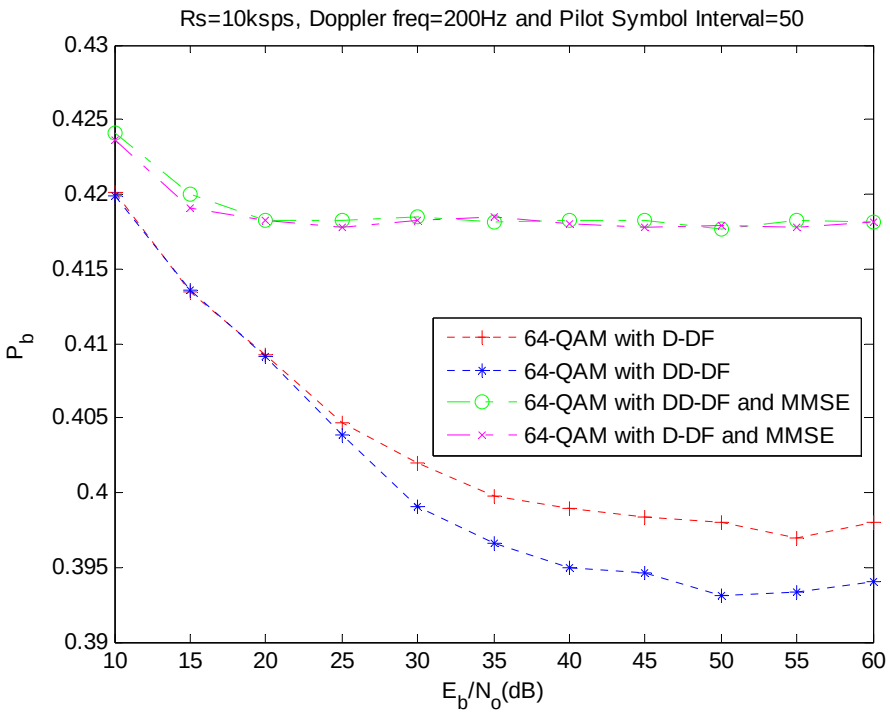


Figure 39. BER of 64-QAM with 200 Hz Doppler Shift and Pilot Symbol Interval of 50.

D. ADAPTIVE TOLERANCES TO DETECT SURGES IN CHANNEL ESTIMATION

1. Effectiveness of Adaptive Tolerances for Low Doppler Frequency Shifts

As mentioned previously, as E_b/N_o increases, the channel estimate becomes less noisy, resulting in lesser variance. Therefore, to prevent the minimum mean square error from over-estimating and compensating for the phase error, the tolerances can be reduced accordingly. One of the means to adaptively adjust the tolerances is to make use of the variance of the channel tap phase errors within the frame. The use of this methodology dramatically reduces the bit error rates for higher E_b/N_o, as seen in Figure 40. It also reduces the bit error rates slightly for the lower E_b/N_o levels. The tighter the tolerances, the better the performance. A tolerance factor is simply the factor that is multiplied by the

variances. In other words, if the variance of the channel tap estimate is σ_h, then the tolerance used would be $k\sigma_h$, where k is the adaptive tolerance factor. As seen in Figure 40, there is marginal benefit in reducing the adaptive tolerance factor below 0.1.

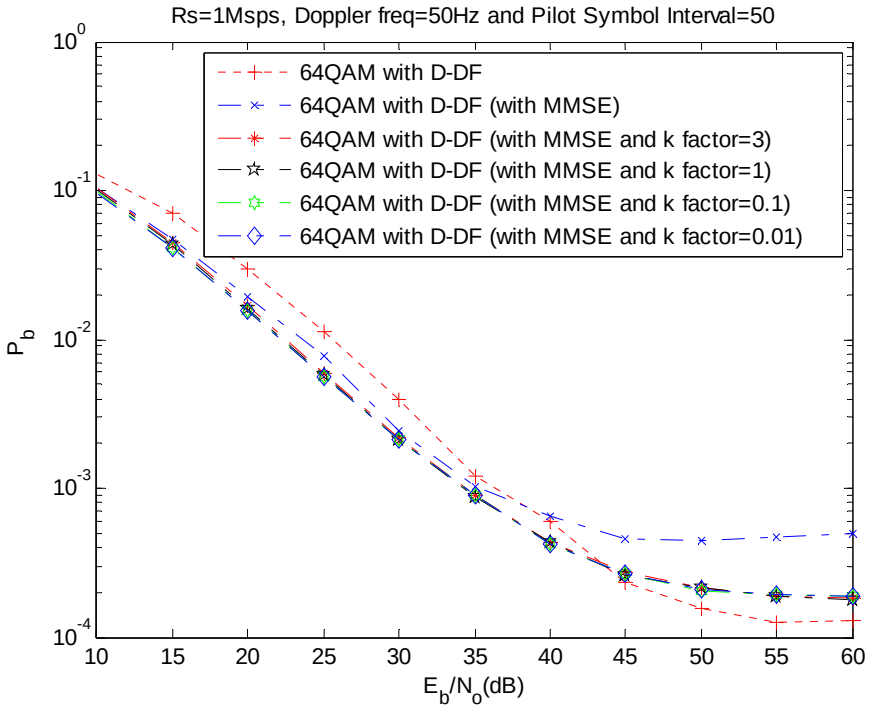


Figure 40. 64-QAM with D-DF with MMSE Filter with Various Adaptive Tolerance Factors.

With the double-differential decision-feedback with a minimum mean square error filter, the implementation of adaptive tolerance worked well at lower E_b/N_o levels but was not effective at higher E_b/N_o levels as shown in Figure 41.

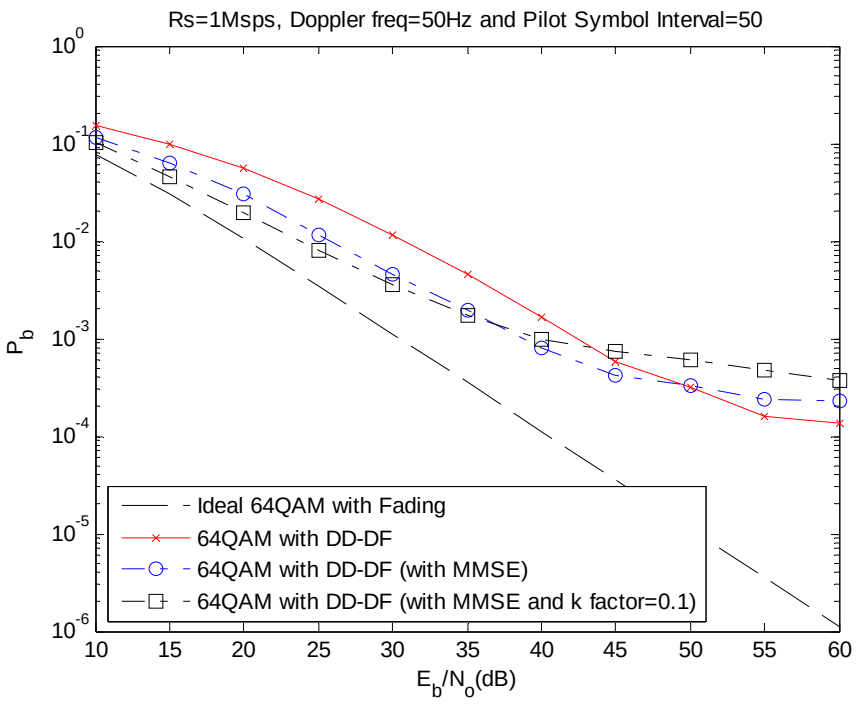


Figure 41. 64-QAM with DD-DF with MMSE Filter with Adaptive Tolerance Factor of 0.1.

Extending the adaptive tolerance to 256-QAM, we find that the performance of the MMSE with adaptive tolerance is better than the algorithm with just differential decision-feedback, as shown in Figure 42. On the other hand, similar to the case for 64-QAM, the implementation of adaptive tolerance onto the double-differential decision-feedback algorithm is not ideal. The bit error rate performance is similar to the case with differential decision-feedback at low E_b/N_o but is much worse at high E_b/N_o as shown in Figure 43. In fact, it is worse than the performance of double-differential decision-feedback with the minimum mean square error algorithm.

Therefore, in summary, the adaptive tolerance approach is only effective for the differential decision-feedback with minimum mean square error algorithm. When it is applied to the double-differential decision-feedback with minimum mean square error algorithm, the bit error rate is worse than having a fixed tolerance of 0.1. It was also

observed that with tighter tolerance factors such as 0.01, the bit error rate decreases, as shown in Figure 44, and does not show major improvement as compared to tolerance factor of 0.1.

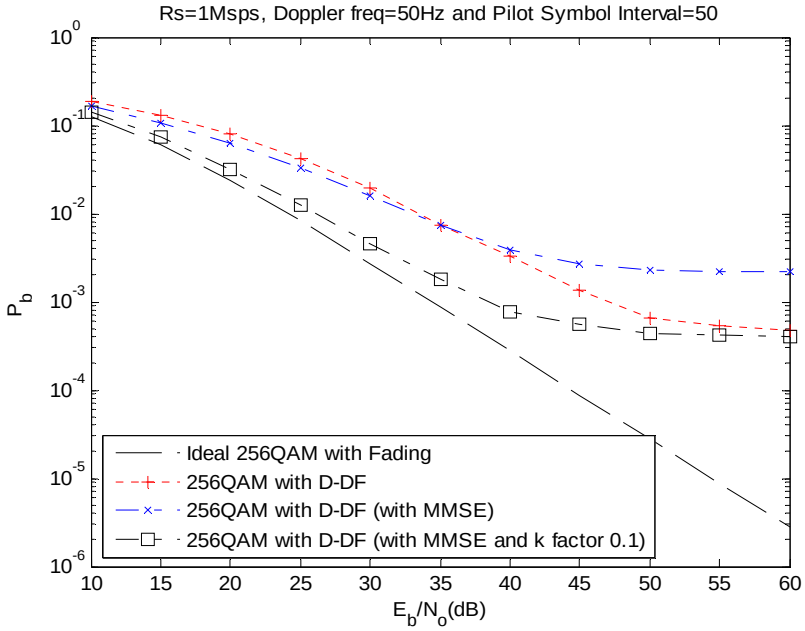


Figure 42. 256-QAM with D-DF with MMSE Filter with Adaptive Tolerance Factor of 0.1.

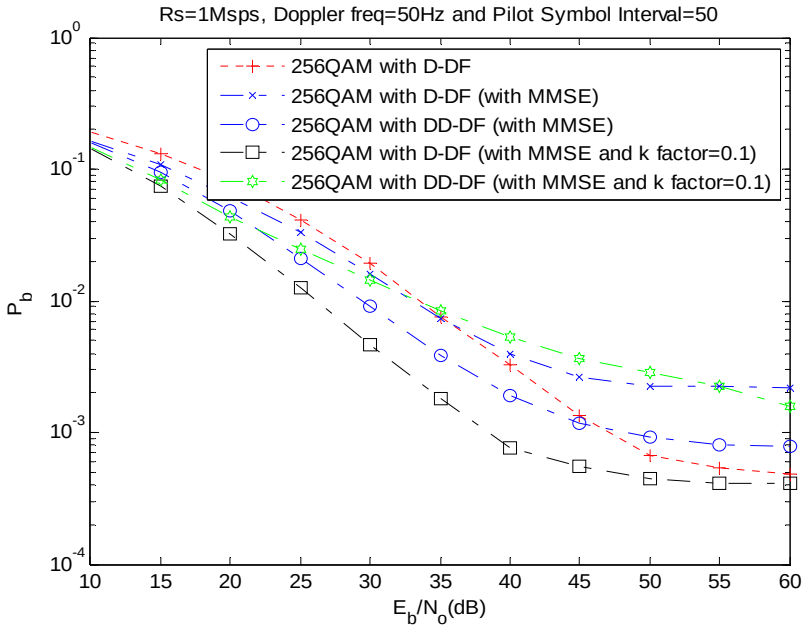


Figure 43. 256-QAM with DD-DF with MMSE Filter with Adaptive Tolerance Factor of 0.1.

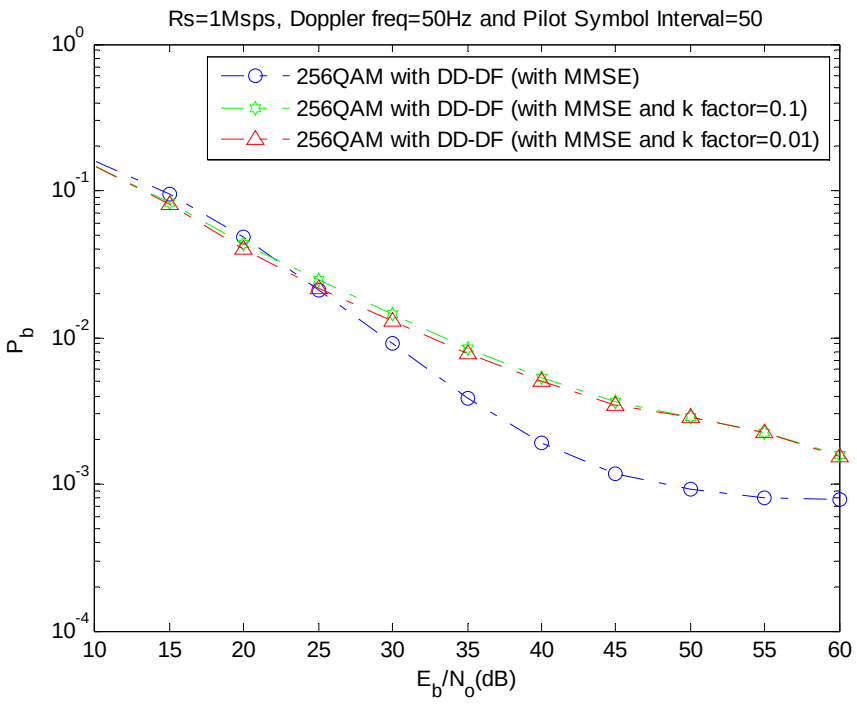


Figure 44. 256-QAM with DD-DF with MMSE Filter with Adaptive Tolerance Factor of 0.01.

With 512-QAM and 1024-QAM, the adaptive tolerance is also very effective in reducing the bit error rate performance throughout the entire range of E_b/N_o, except for the higher E_b/N_o on 1024-QAM modulation. The effectiveness of adaptive tolerance decreases at higher E_b/N_o for higher modulations, as can be seen in Figures 45 and 46.

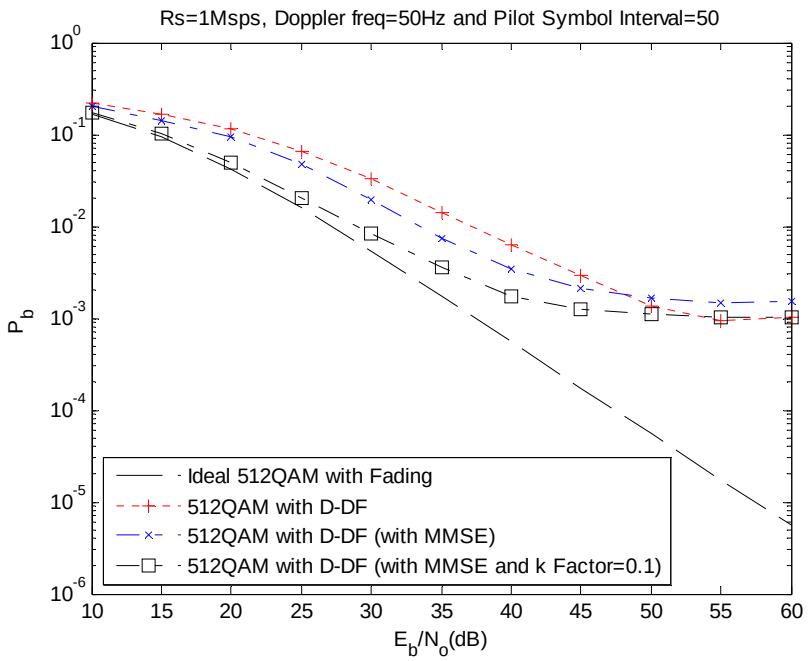


Figure 45. 512-QAM with D-DF with MMSE Filter with Adaptive Tolerance Factor of 0.1.

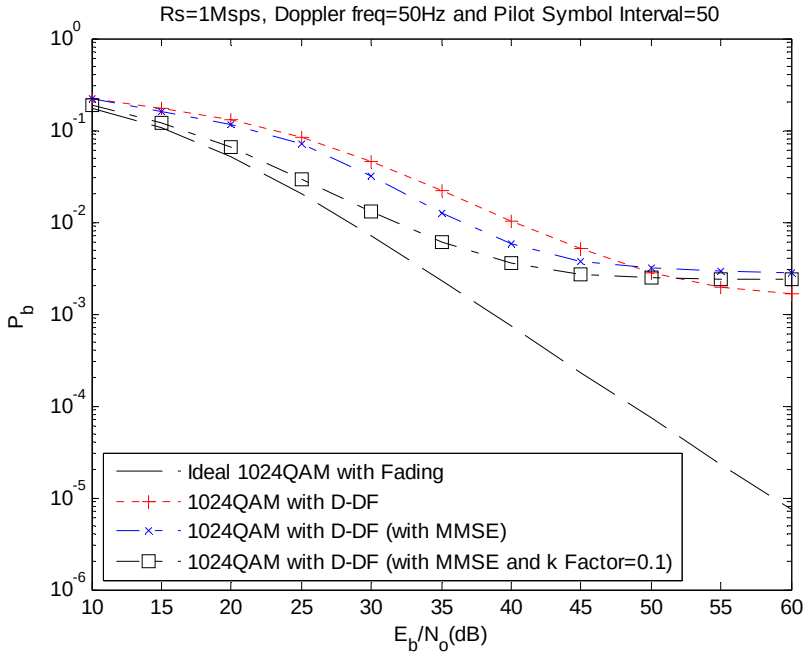


Figure 46. 1024-QAM with D-DF with MMSE Filter with Adaptive Tolerance Factor of 0.1.

2. Comparison of the Minimum Mean Square Error Implementation on Differential Decision-Feedback and Double-Differential Decision-Feedback Algorithm With Adaptive Tolerances

To understand the effectiveness of adaptive tolerance on the various algorithms, the effectiveness of adaptive tolerances on the differential decision-feedback and double-differential decision-feedback algorithms with minimum mean square error are shown in Tables 5 to 7. When the use of adaptive tolerance improved the bit error rate performance, it is labeled as 'Y*', meaning it is effective with improvement. If it caused the performance to decrease, it is labeled as 'N*', meaning that is not effective and caused performance degradation. The use of adaptive tolerance improved the minimum mean square error algorithm for the differential decision-feedback algorithm with minimum mean square error but reduced the performance of the double-differential decision-feedback with minimum mean square error algorithm. The update of the most effective algorithm for the 64-QAM to 1024-QAM is shown in Table 7. The use of adaptive tolerance improved the overall performance of the differential decision-feedback with minimum mean square error algorithm, especially for 256-QAM and 512-QAM, making it the most effective algorithm for all SNR.

Table 5. Tabulation of Adaptive Tolerance Effectiveness on D-DF with MMSE (Y= Effective; Y*=Effective with Improvement; N= Not Effective).

SNR (dB)	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**
64-QAM D-DF	Y	Y	Y	Y	Y	Y	Y*	N	N	N	N
256-QAM D-DF	Y	Y	Y	Y	Y	Y	Y	Y	Y*	Y*	Y*
512-QAM D-DF	Y	Y	Y	Y	Y	Y	Y	Y	Y*	Y*	Y*
1024-QAM D-DF	Y	Y	Y	Y	Y	Y	Y	Y	Y*	N	N

Table 6. Tabulation of Adaptive Tolerance Effectiveness on DD-DF with MMSE (Y= Effective; N= Not Effective; N*=Not Effective with Degradation).

SNR (dB)	10	15	20	25	30	35	40	45	50	55	60
64-QAM	Y	Y	Y	Y	Y	Y	Y	N*	N	N	N
256-QAM	Y	Y	Y	Y	Y	N*	N*	N*	N*	N*	N*

Table 7. Tabulation of Most Effective Algorithms for Different Modulations.

Modulation	Most Effective Algorithm (low SNR)	Most Effective Algorithm (High SNR)
64-QAM	D-DF with MMSE/Adaptive Tol	D-DF
256-QAM	D-DF with MMSE/Adaptive Tol	D-DF with MMSE/Adaptive Tol
512-QAM	D-DF with MMSE/Adaptive Tol	D-DF with MMSE/Adaptive Tol
1024-QAM	D-DF with MMSE/Adaptive Tol	D-DF

3. Effectiveness of Adaptive Tolerances for High Doppler Frequency Shifts

As with the high Doppler frequency shifts of 100 Hz and 200 Hz, the performance of 64-QAM with a symbol rate of 100 ksps was obtained. The pilot symbol intervals used were 12, 25 and 50 symbols.

For a Doppler frequency shift of 100 Hz, as can be seen from Figures 47 to 50, although the adaptive tolerance did improve the bit error rate performance of the differential decision-feedback with minimum mean square error algorithm, the improvement was not sufficient to compensate for the degradation caused by the Minimum Mean Square algorithm. In all cases of different pilot symbol intervals, the use of adaptive tolerance helped improve the bit error rate performances of both the differential decision-feedback with minimum mean square error and double-differential decision-feedback with minimum mean square error algorithms.

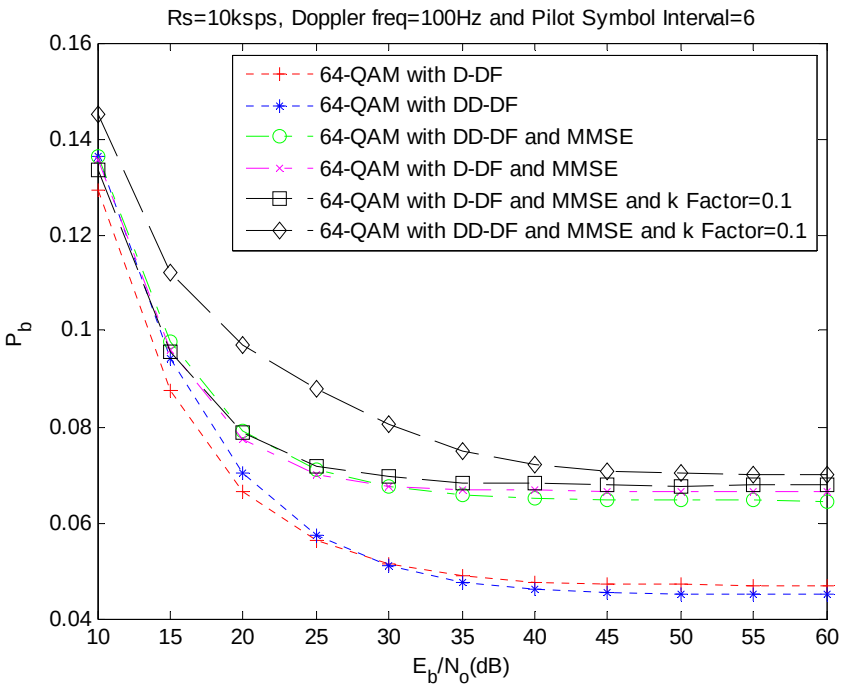


Figure 47. 64-QAM with 100 Hz Doppler Frequency Shift, Pilot Symbol Interval of 6 and Adaptive Tolerance Factor of 0.1.

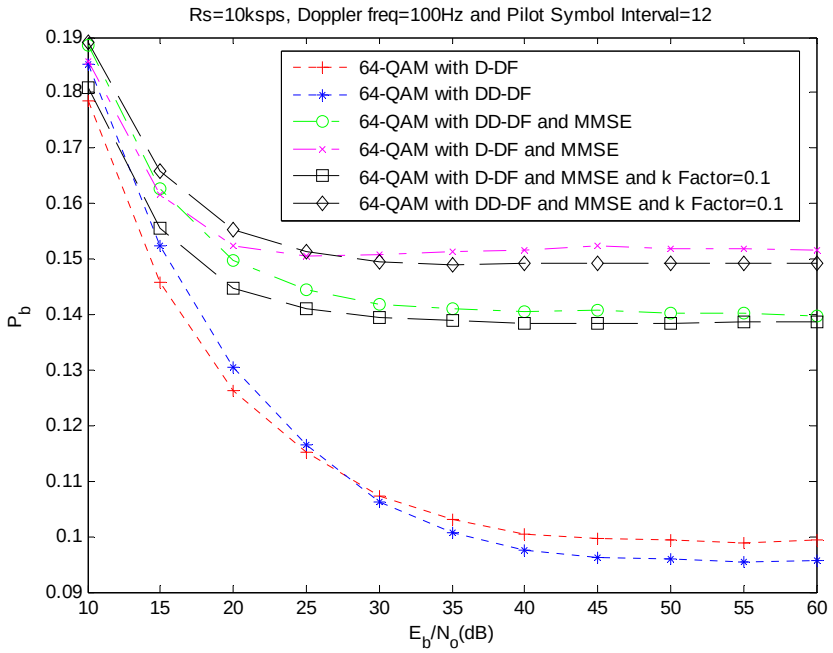


Figure 48. 64-QAM with 100 Hz Doppler Frequency Shift, Pilot Symbol Interval of 12 and Adaptive Tolerance Factor of 0.1.

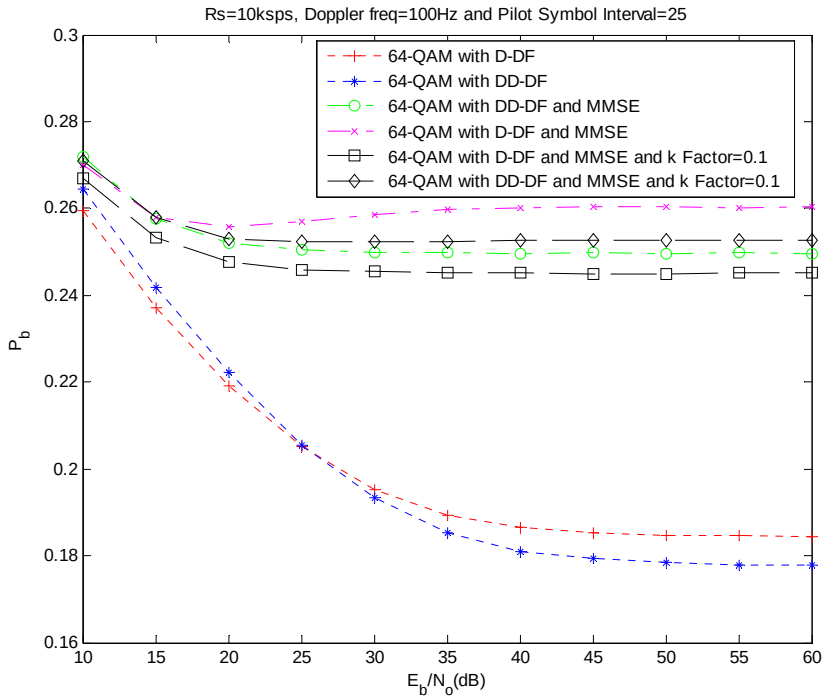


Figure 49. 64-QAM with 100 Hz Doppler Frequency Shift, Pilot Symbol Interval of 25 and Adaptive Tolerance Factor of 0.1.

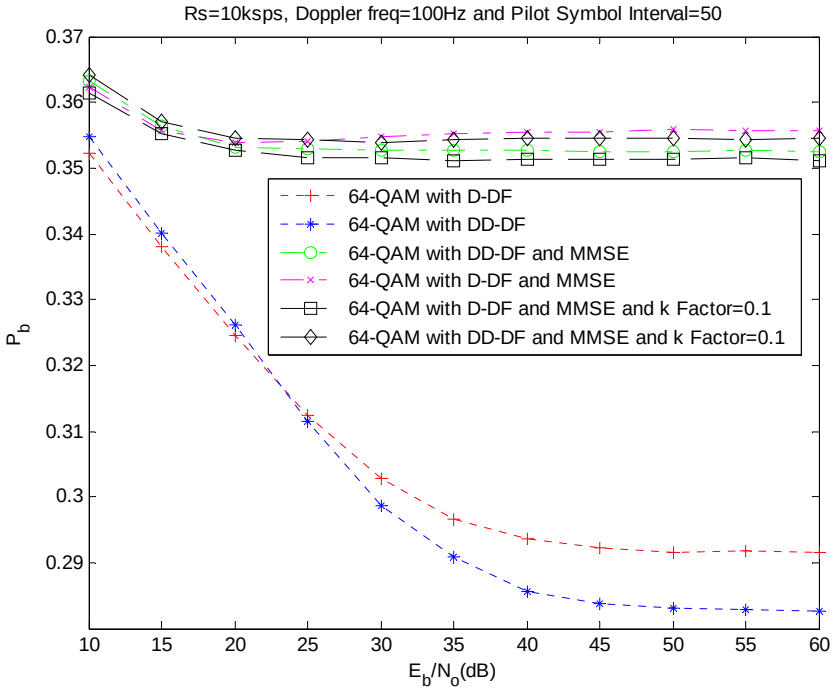


Figure 50. 64-QAM with 100 Hz Doppler Frequency Shift, Pilot Symbol Interval of 50 and Adaptive Tolerance Factor of 0.1.

As for the case with 200 Hz Doppler frequency shift, the adaptive tolerance was only able to improve the differential decision-feedback with minimum mean square error algorithm. The use of adaptive tolerance on the double-differential decision-feedback with minimum mean square error algorithm degraded the bit error rate performances for all different values of pilot symbol interval. At this high Doppler frequency shift, the channel is fast fading and rapidly changing with respect to the pilot symbol interval; hence, the use of adaptive tolerance may be too restrictive, rejecting several legitimate changes to the channel estimate. The bit error rate performances of 64-QAM with pilot symbol intervals of 6, 12, 25 and 50 symbols for all four different algorithms, differential decision-feedback and double-differential decision-feedback, with and without minimum mean square error, are shown in Figures 51 to 54.

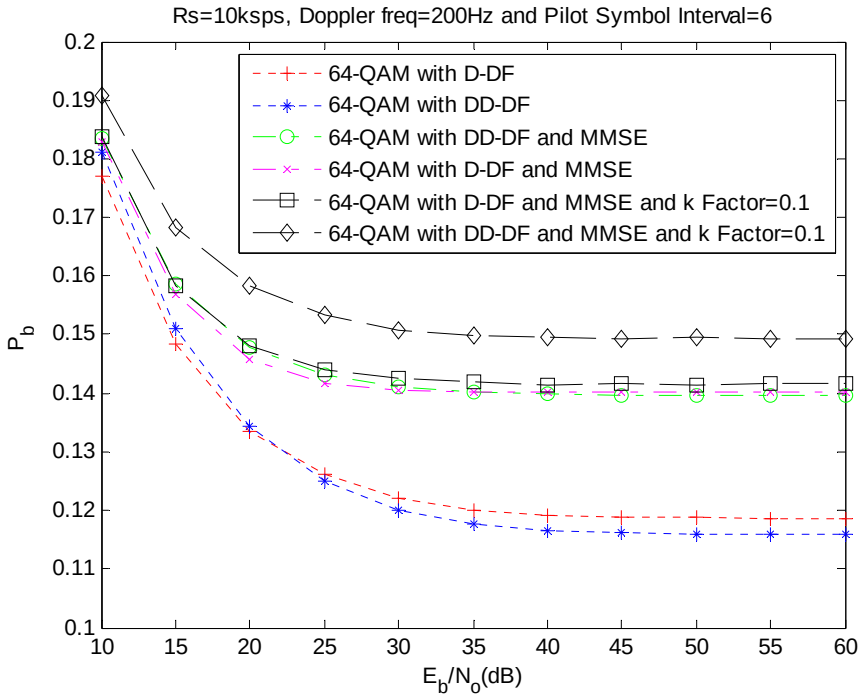


Figure 51. 64-QAM with 200 Hz Doppler Frequency Shift, Pilot Symbol Interval of 6 and Adaptive Tolerance Factor of 0.1.

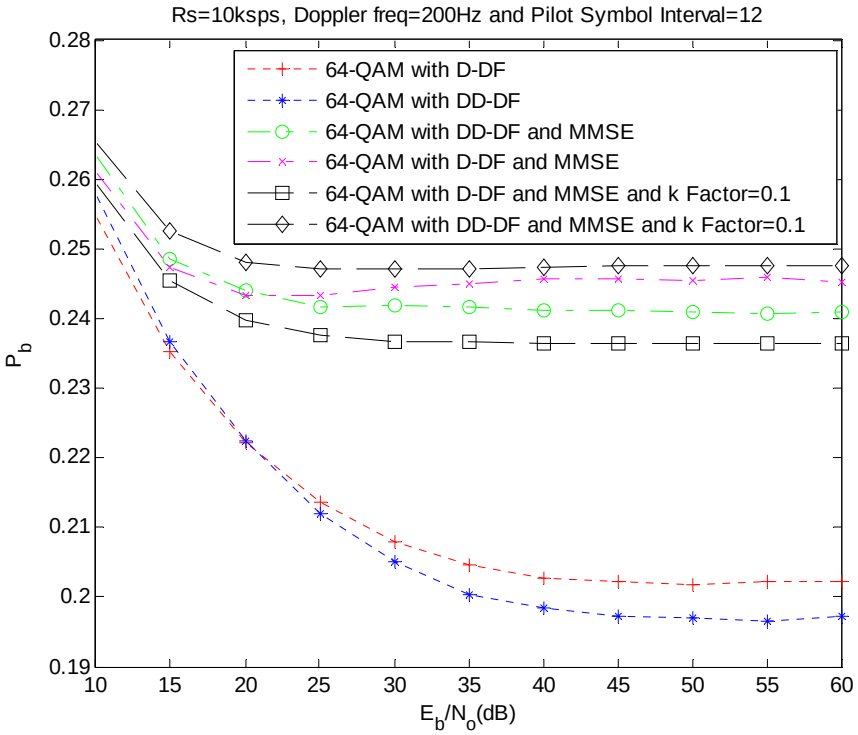


Figure 52. 64-QAM with 200 Hz Doppler Frequency Shift, Pilot Symbol Interval of 12 and Tolerance Factor of 0.1.

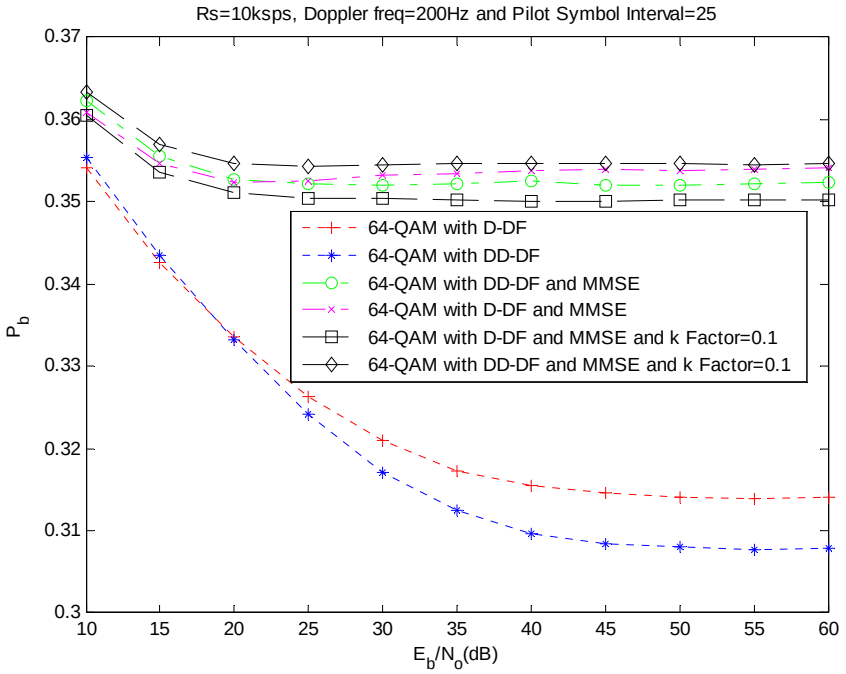


Figure 53. 64-QAM with 200 Hz Doppler Frequency Shift, Pilot Symbol Interval of 25 and Tolerance Factor of 0.1.

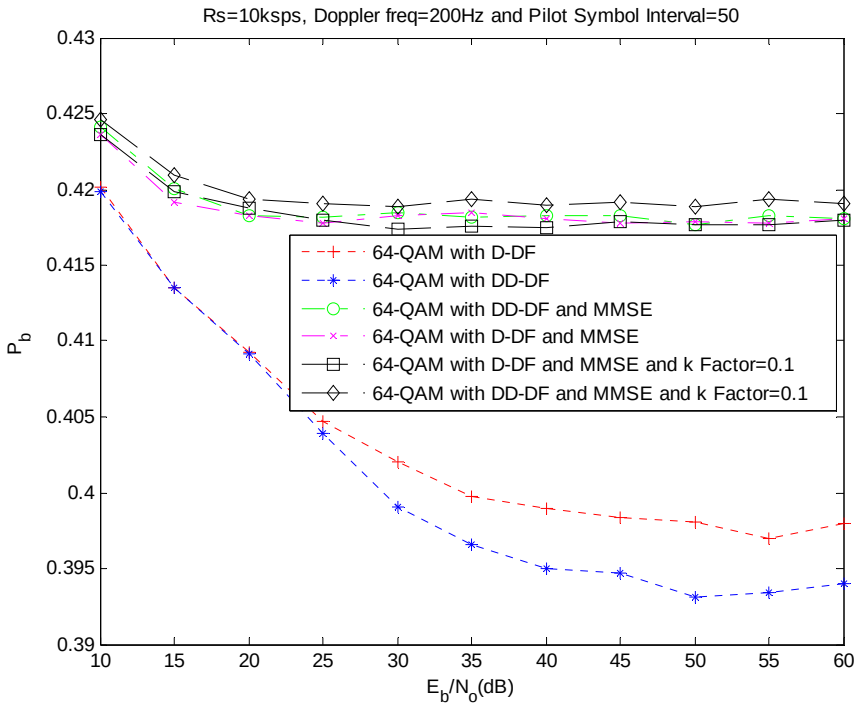


Figure 54. 64-QAM with 200 Hz Doppler Frequency Shift, Pilot Symbol Interval of 50 and Tolerance Factor of 0.1.

IV. CONCLUSION AND RECOMMENDATIONS

A. CONCLUSION

The previous chapters show that the differential decision-feedback and double-differential decision-feedback algorithms are already very effective in estimating the channel tap phase errors for each symbol. With the inclusion of minimum mean square error filtering, the new algorithm does improve the bit error rate performances, especially at lower E_b/N_o. Although the performance is greatly improved at lower E_b/N_o, the minimum mean square error algorithm tends to introduce some errors at the higher E_b/N_o, causing the bit error rate performances to taper off at a fixed value. At higher E_b/N_o levels, the basic differential decision-feedback algorithm is the most effective.

The implementation of minimum mean square error estimation also dramatically improved the bit error rate performances when used with the double-differential decision-feedback algorithm. This combination of the double-differential decision-feedback algorithm with minimum mean square error estimation provided the best performance at low E_b/N_o for higher order modulations, such as 256-QAM and above. Otherwise, for lower order modulations, the differential decision-feedback with minimum mean square error algorithm provides the best performance.

For cases with high Doppler frequency shifts, the most effective algorithm is double-differential decision-feedback. The use of minimum mean square error degraded the performance and resulted in higher bit error rates than the simpler differential decision-feedback and double-differential decision-feedback algorithms. This was expected since the correlation between channel taps is low and using past channel tap estimates to determine the current channel tap estimate tends to introduce errors.

The use of adaptive tolerances, based on the channel estimate's variance, proved to be effective in improving the bit error rate performances for the differential decision-feedback with minimum mean square error algorithm. When used in conjunction with the

differential decision-feedback with minimum mean square error algorithm, it is generally the most effective for all modulation schemes. The use of adaptive tolerances is, however, not useful for the double-differential decision-feedback with minimum mean square error algorithm as it tends to introduce many errors at high E_b/N_o levels. It is also worse than the performance of adaptive tolerance used with the differential decision-feedback with minimum mean square error algorithm.

B. RECOMMENDATIONS

Studies on different error correction coding schemes implemented with the algorithms introduced in this thesis could provide insight on the performance of such algorithms in practical systems.

Another area of studies is to extend these algorithms to multiple-input multiple-output (MIMO) communication systems.

LIST OF REFERENCES

[1] T. Ha. Theory and Design of Digital Communication Systems. Cambridge University Press, 2011.

[2] Jeffrey A. Smith. Doppler Effects on Single-Carrier Signals Operating in Fading Channels. Thesis, Naval Postgraduate School, 2009.

[3] Charles W. Therrien. Discrete Random Signals and Statistical Signal Processing. Prentice Hall Signal Processing Series, 2004.

[4] Dimitris G. Manolakis, Vinay K. Ingle and Stephen M. Kogon. Statistical and Adaptive Signal Processing. Artech Hose Signal Processing Library, 2005.

[5] K. Sam Shanmugan and A. M. Breiphol. Random Signals, Detection, Estimation and Data Analysis. John Wiley & Son, 1988.

[6] Robert G. Brown. Introduction to Random Signal Analysis and Kalman Filtering. John Wiley & Son, 1983.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A - MINIMUM MEAN SQUARE ERROR

A. GENERAL DERIVATION [5]

Given the observation $x(1), \ldots, x(n)$, the original signal, $s = x + \varepsilon$, where ε is the error between the original signal and the observation, could be estimated by $\hat{s}$ with the linear estimator of the form $\hat{s} = h_0 + \sum_{i=1}^{n} h_i x(i)$ such that $\mathrm{E}\left\{\left(s - \hat{s}\right)^2\right\}$ is minimized by the choice of the h_i for $i = 1, \ldots, n$.

Differentiating $\mathrm{E}\left\{\left(s - \hat{s}\right)\right\}$ with respect to each h_j for $j = 1, \ldots, n$, results in

$$\begin{aligned} &\mathrm{E}\left\{\left(s - h_0 - \sum_{i=1}^{n} h_i x(i)\right)\right\} = 0 \\ &\mathrm{E}\left\{x(j)\left(s - h_0 - \sum_{i=1}^{n} h_i x(i)\right)^*\right\} = 0, \qquad j = 1, \ldots, n \end{aligned} \tag{A.1}$$

These two conditions are called the *orthogonality conditions*. Simplifying both equations, we obtain the following:

$$h_0 = \mu_s - \sum_{i=1}^{n} h_i \mu_{x(i)} \tag{A.2}$$

$$\sum_{i=1}^{n} h_i C_x(i, j) = \sigma^*_{sx(j)}, \quad j = 1, 2, \ldots, n \tag{A.3}$$

where

$$C_x(i, j) = \mathrm{E}\left\{\left[x(i) - \mu_{x(i)}\right]\left[x(j) - \mu_{x(j)}\right]\right\}$$

$$\sigma_{sx(j)} = \mathrm{E}\left\{\left[s - \mu_s\right]\left[x(j) - \mu_{x(j)}\right]\right\}$$

B. APPLICATION TO CHANNEL TAP ESTIMATION

Since the desired estimator, $\hat{h}$, is given by the relationship $\hat{h} = \tilde{h} + \varepsilon$, where $\tilde{h}$ is the observation, the following is obtained by replacing $C_x(i, j)$ with $R_{\hat{h}}$ and $\sigma^*_{sx(j)}$ with $\underset{\tilde{h}\hat{h}}{\boldsymbol{r}}$ in Equation (A.3). Letting $\boldsymbol{h}_{opt} = \sum_{i=1}^{n} h_i$, the new channel tap estimate, $\boldsymbol{h}_{est}$ could be obtained through the following Equations (A.4) and (A.5)

$$R_{\hat{h}} \boldsymbol{h}_{opt} = \underset{\tilde{h}\hat{h}}{\boldsymbol{r}}^* \tag{A.4}$$

$$h_0 = \mu_{\tilde{h}} - \boldsymbol{h}_{opt} \mu_{\hat{h}} \tag{A.5}$$

$$\boldsymbol{h}_{est} = \sum_{i=1}^{n} h_i \hat{\boldsymbol{h}} + h_0 = \boldsymbol{h}_{opt} \hat{\boldsymbol{h}} + \mu_{\tilde{h}} - \boldsymbol{h}_{opt} \mu_{\hat{h}} \tag{A.6}$$

Combining the equations as shown above, the final minimum mean square estimator for the channel tap is $\boldsymbol{h}_{est} = \boldsymbol{h}_{opt} \hat{\boldsymbol{h}} + \mu_{\tilde{h}} - \boldsymbol{h}_{opt} \mu_{\hat{h}}$ (as shown in Equation (7) in Chapter III).

APPENDIX B - MATLAB SOURCE CODES

A. DIFFERENTIAL DECISION-FEEDBACK AND DOUBLE-DIFFERENTIAL DECISION-FEEDBACK

1. Differential Decision-Feedback

```
function r = D_DF_demod(m, hmod, hdemod, pilotSymbol, pilotInterval)
% edited from Jeff Smith
% NPS
% Demodulates using D-DF algorithm

Tol = .1;
r = zeros(1,length(m));     % represents the demodulated received message
zpilot = modulate(hmod,pilotSymbol);
for n = 1:length(m)
   if rem(n,pilotInterval) == 1    % identify the pilot symbols
      r(n) = pilotSymbol;        % let received symbol be pilot symbol
      h = m(n) / zpilot;         % determine channel tap
      e = angle(h);           % calculate channel tap phase error
      e1 = 0;
      a = abs(h);              % calculate channel tap magnitude
      f(n) = h;
   else
      r(n) = demodulate(hdemod,m(n)/a*conj(h)/abs(h)*exp(-1i*e1));
      z = modulate(hmod,r(n));                        % calculate new channel tap
      h1 = m(n) / z;
      eh = -e+angle(h1);
      if   or(abs(eh-e1)<   Tol,   and(and(abs(eh)>(pi-Tol),abs(e1)>(pi-Tol)),or(abs(eh-e1-
2*pi)< Tol,abs(eh-e1+2*pi)< Tol)))
         e1 = eh;
      end
    end
end
end
```

2. Double-Differential Decision-Feedback

```
function r = DD_DF_demod(m, hmod, hdemod, pilotSymbol, pilotInterval)
% adapted from Jeff Smith
% NPS
% Demodulates using DD-DF algorithm

Tol = .1;
r = zeros(1,length(m));     % represents the demodulated received message
zpilot = modulate(hmod,pilotSymbol);
for n = 1:length(m)
   if rem(n,pilotInterval) == 1    % identify the pilot symbols
      r(n) = pilotSymbol;        % let received symbol be pilot symbol
      h = m(n) / zpilot;         % determine channel tap
      e = angle(h);           % calculate channel tap phase error
      e1 = 0;
      e2 = 0;
      a = abs(h);               % calculate channel tap magnitude
   else
      r(n) = demodulate(hdemod,m(n)/a*conj(h)/abs(h)*exp(-1i*(2*e1-e2)));
      z = modulate(hmod,r(n));                        % calculate new channel tap
      h1 = m(n) / z;
      eh = -e+angle(m(n)/z);
      if or(((abs(eh-e1))< Tol),and(and(abs(e1+e)>(pi-Tol),abs(eh+e)>(pi-Tol)),or(abs(eh-
e1-2*pi)< Tol,abs(eh-e1+2*pi)< Tol)))
         e2 = e1;
         e1 = eh;
      end
   end
end
```

B. DIFFERENTIAL DECISION-FEEDBACK AND DOUBLE-DIFFERENTIAL DECISION-FEEDBACK WITH MINIMUM MEAN SQUARE ERROR

1. Differential Decision-Feedback With MMSE

```
function r = D_DF_demod_MMSE_AT(m, hmod, hdemod, pilotSymbol, pilotInterval,
Rs)
% Written by Gabriel Tham
% Used parts of the code from Jeff Smith
% NPS
% Demodulates using D-DF algorithm with MMSE

Tol = .1;
r = zeros(1,length(m));     % represents the demodulated received message
f = zeros(1,length(m));     % records all phase errors
min_filter_length = 3;
zpilot = modulate(hmod,pilotSymbol);
PS_Num = 0;
for n = 1:length(m)
   if and(PS_Num >=1, rem(n,pilotInterval) == 1)
      Tol2 = Tol;
      hopt2 = MMSE(f(n-pilotInterval:n-1), min_filter_length-1, Rs);
      h_upd=(f(n-pilotInterval)+f(n))/2-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval:n-
pilotInterval+1)) + hopt2(1)*f(n-pilotInterval+1) + hopt2(2)*f(n-pilotInterval);
      r_temp = demodulate(hdemod,m(n-pilotInterval+1)/a*conj(h_upd)/abs(h_upd));
      f(n-pilotInterval+1) = h_upd;
      z_temp = modulate(hmod,r_temp);
      h_temp = m(n-pilotInterval+1)/z_temp;
      if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
         r(n-pilotInterval+1) = r_temp;
         f(n-pilotInterval+1) = h_upd;
         f(n-pilotInterval+2) = h_temp;
      end
      hopt = MMSE(f(n-pilotInterval:n-1), min_filter_length, Rs);
      for i = min_filter_length:pilotInterval
         h_upd =(f(n-pilotInterval)+f(n))/2-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-1)) + hopt(1)*f(n-pilotInterval+i-1) + hopt(2)*f(n-
pilotInterval+i-2) + hopt(3)*f(n-pilotInterval+i-3);
         r_temp = demodulate(hdemod,m(n-pilotInterval+i-1)/a*conj(h_upd)/abs(h_upd));
         z_temp = modulate(hmod,r_temp);
         h_temp = m(n-pilotInterval+i-1)/z_temp;
         if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
            r(n-pilotInterval+i-1) = r_temp;
```

```
                    f(n-pilotInterval+i-1) = h_upd;
                    f(n-pilotInterval+i) = h_temp;      % update the next symbol channel with the
latest estimate (may take out later)
                 else if n-pilotInterval+i-1 == 3
                    h_upd =(f(n-pilotInterval)+f(n))/2-(hopt2(1)+hopt2(2))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-2)) + hopt2(1)*f(n-pilotInterval+i-2) + hopt2(2)*f(n-
pilotInterval+i-3);
                    r_temp = demodulate(hdemod,m(n-pilotInterval+i-
1)/a*conj(h_upd)/abs(h_upd));
                    z_temp = modulate(hmod,r_temp);
                    h_temp = m(n-pilotInterval+i-1)/z_temp;
                    r(n-pilotInterval+i-1) = r_temp;
                    f(n-pilotInterval+i) = h_temp;
                    f(n-pilotInterval+i-1) = h_upd;
                    else
                       h_upd =(f(n-pilotInterval)+f(n))/2-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-4:n-pilotInterval+i-2)) + hopt(1)*f(n-pilotInterval+i-2) + hopt(2)*f(n-
pilotInterval+i-3) + hopt(3)*f(n-pilotInterval+i-4);
                       r_temp = demodulate(hdemod,m(n-pilotInterval+i-
1)/a*conj(h_upd)/abs(h_upd));
                       z_temp = modulate(hmod,r_temp);
                       h_temp = m(n-pilotInterval+i-1)/z_temp;
                       r(n-pilotInterval+i-1) = r_temp;
                       f(n-pilotInterval+i) = h_temp;
                       f(n-pilotInterval+i-1) = h_upd;
                    end
                 end
              end
           end
           if rem(n,pilotInterval) == 1    % identify the pilot symbols
              r(n) = pilotSymbol;        % let received symbol be pilot symbol
              h = m(n) / zpilot;         % determine channel tap
              e = angle(h);           % calculate channel tap phase error
              e1 = 0;
              a = abs(h);               % calculate channel tap magnitude
              pilot_channel = h;
              f(n) = pilot_channel;
              original_chan(n) = f(n);
              PS_Num = PS_Num + 1;
           else
              r(n) = demodulate(hdemod,m(n)/a*conj(h)/abs(h)*exp(-1i*e1));
              z = modulate(hmod,r(n));                        % calculate new channel tap
              h1 = m(n) / z;
              eh = -e+angle(h1);
              if or(abs(eh-e1)< Tol, and(and(abs(eh)>(pi-Tol),abs(e1)>(pi-Tol)),or(abs(eh-e1-
2*pi)< Tol,abs(eh-e1+2*pi)< Tol)))
                 f(n) = h1;
                 e1 = eh;
              else
                 f(n) = h1;
```

```
            end
          end
        end
        if mod(n,pilotInterval)~= 0
          pilotInterval = mod(n,pilotInterval);
        end
            Tol2 = min(Tol_factor*var(f(n-pilotInterval:n-1)),Tol);
            hopt2 = MMSE(f(n-pilotInterval+1:n), min_filter_length-1, Rs);
            h_upd = f(n-pilotInterval+1)-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval+1:n-
pilotInterval+2)) + hopt2(1)*f(n-pilotInterval+2) + hopt2(2)*f(n-pilotInterval+1);
            r_temp = demodulate(hdemod,m(n-pilotInterval+2)/a*conj(h_upd)/abs(h_upd));
            f(n-pilotInterval+2) = h_upd;
            z_temp = modulate(hmod,r_temp);
            h_temp = m(n-pilotInterval+2)/z_temp;
            if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
              r(n-pilotInterval+2) = r_temp;
              f(n-pilotInterval+2) = h_upd;
              f(n-pilotInterval+3) = h_temp;
            end
            hopt = MMSE(f(n-pilotInterval+1:n), min_filter_length, Rs);
            for i = min_filter_length:pilotInterval
              h_upd =f(n-pilotInterval)-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-pilotInterval+i-
2:n-pilotInterval+i)) + hopt(1)*f(n-pilotInterval+i) + hopt(2)*f(n-pilotInterval+i-1) + hopt(3)*f(n-
pilotInterval+i-2);
              r_temp = demodulate(hdemod,m(n-pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
              z_temp = modulate(hmod,r_temp);
              h_temp = m(n-pilotInterval+i)/z_temp;
              if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
                r(n-pilotInterval+i) = r_temp;
                f(n-pilotInterval+i) = h_upd;
                f(n-pilotInterval+i+1) = h_temp;
              else if n-pilotInterval+i == 3
                h_upd = f(n-pilotInterval)-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval+i-2:n-
pilotInterval+i-1)) + hopt2(1)*f(n-pilotInterval+i-1) + hopt2(2)*f(n-pilotInterval+i-2);
                r_temp = demodulate(hdemod,m(n-pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
                z_temp = modulate(hmod,r_temp);
                h_temp = m(n-pilotInterval+i)/z_temp;
                r(n-pilotInterval+i) = r_temp;
                f(n-pilotInterval+i+1) = h_temp;
                f(n-pilotInterval+i) = h_upd;
                else
                  h_upd = f(n-pilotInterval)-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-1)) + hopt(1)*f(n-pilotInterval+i-1) + hopt(2)*f(n-
pilotInterval+i-2) + hopt(3)*f(n-pilotInterval+i-3);
                  r_temp = demodulate(hdemod,m(n-
pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
```

```
                z_temp = modulate(hmod,r_temp);
                h_temp = m(n-pilotInterval+i)/z_temp;
                r(n-pilotInterval+i) = r_temp;
                f(n-pilotInterval+i+1) = h_temp;
                f(n-pilotInterval+i) = h_upd;
              end
            end
          end
        end
```

2. Double-Differential Decision-Feedback With MMSE

```
function r = DD_DF_demod_MMSE(m, hmod, hdemod, pilotSymbol, pilotInterval, Rs)
% Written by Gabriel Tham
% Used parts of the code from Jeff Smith
% NPS
% Demodulates using DD-DF algorithm with MMSE

Tol = .1;
r = zeros(1,length(m));     % represents the demodulated received message
f = zeros(3,length(m));     % records all phase errors
min_filter_length = 3;
zpilot = modulate(hmod,pilotSymbol);
PS_Num = 0;
for n = 1:length(m)
   if and(PS_Num >=1, rem(n,pilotInterval) == 1)
      Tol2 = Tol;
      hopt2 = MMSE(f(n-pilotInterval:n-1), min_filter_length-1, Rs);
      h_upd =(f(n-pilotInterval)+f(n))/2-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval:n-
pilotInterval+1)) + hopt2(1)*f(n-pilotInterval+1) + hopt2(2)*f(n-pilotInterval);
      r_temp = demodulate(hdemod,m(n-pilotInterval+1)/a*conj(h_upd)/abs(h_upd));
      z_temp = modulate(hmod,r_temp);
      h_temp = m(n-pilotInterval+1)/z_temp;
      if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
         r(n-pilotInterval+1) = r_temp;
         f(n-pilotInterval+1) = h_temp;
      end
      hopt = MMSE(f(n-pilotInterval:n-1), min_filter_length, Rs);
      for i = min_filter_length:pilotInterval
         h_upd =(f(n-pilotInterval)+f(n))/2-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-1)) + hopt(1)*f(n-pilotInterval+i-1) + hopt(2)*f(n-
pilotInterval+i-2) + hopt(3)*f(n-pilotInterval+i-3);
         r_temp = demodulate(hdemod,m(n-pilotInterval+i-1)/a*conj(h_upd)/abs(h_upd));
         z_temp = modulate(hmod,r_temp);
         h_temp = m(n-pilotInterval+i-1)/z_temp;
```

```
                if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
                    r(n-pilotInterval+i-1) = r_temp;
                    f(n-pilotInterval+i) = h_temp;      % update the next symbol channel with the
latest estimate (may take out later)
                else if n-pilotInterval+i-1 == 3
                    h_upd =(f(n-pilotInterval)+f(n))/2-(hopt2(1)+hopt2(2))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-2)) + hopt2(1)*f(n-pilotInterval+i-2) + hopt2(2)*f(n-
pilotInterval+i-3);
                    r_temp = demodulate(hdemod,m(n-pilotInterval+i-
1)/a*conj(h_upd)/abs(h_upd));
                    z_temp = modulate(hmod,r_temp);
                    h_temp = m(n-pilotInterval+i-1)/z_temp;
                    r(n-pilotInterval+i-1) = r_temp;
                    f(n-pilotInterval+i) = h_temp;
                    f(n-pilotInterval+i-1) = h_upd;
                    else
                        h_upd =(f(n-pilotInterval)+f(n))/2-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-4:n-pilotInterval+i-2)) + hopt(1)*f(n-pilotInterval+i-2) + hopt(2)*f(n-
pilotInterval+i-3) + hopt(3)*f(n-pilotInterval+i-4);
                        r(n-pilotInterval+i-1) = demodulate(hdemod,m(n-pilotInterval+i-
1)/a*conj(h_upd)/abs(h_upd));
                        z_temp = modulate(hmod,r(n-pilotInterval+i-1));
                        h_temp = m(n-pilotInterval+i-1)/z_temp;
                        f(n-pilotInterval+i) = h_temp;
                        f(n-pilotInterval+i-1) = h_upd;
                    end
                end
            end
            if rem(n,pilotInterval) == 1    % identify the pilot symbols
                r(n) = pilotSymbol;        % let received symbol be pilot symbol
                h = m(n) / zpilot;         % determine channel tap
                e = angle(h);           % calculate channel tap phase error
                e1 = 0;
                e2 = 0;
                a = abs(h);               % calculate channel tap magnitude
                f(n) = h;
                PS_Num = PS_Num + 1;
            else
                r(n) = demodulate(hdemod,m(n)/a*conj(h)/abs(h)*exp(-1i*(2*e1-e2)));    %
demodulate with channel tap info and removed a
                z = modulate(hmod,r(n));                        % calculate new channel tap
                h1 = m(n) / z;
                eh = -e+angle(m(n)/z);
                if or(((abs(eh-e1))< Tol),and(and(abs(e1+e)>(pi-Tol),abs(eh+e)>(pi-Tol)),or(abs(eh-
e1-2*pi)< Tol,abs(eh-e1+2*pi)< Tol)))
                    f(n) = h1;
                    e2 = e1;
```

```
                e1 = eh;
                h=h1;
            else
                f(n) = f(n-1);
             end
          end
       end
       if mod(n,pilotInterval)~= 0
          pilotInterval = mod(n,pilotInterval);
       end
            Tol2 = Tol; %min(10*var(f(n-pilotInterval:n-1)),Tol);
            hopt2 = MMSE(f(n-pilotInterval+1:n), min_filter_length-1, Rs);
            h_upd = f(n-pilotInterval+1)-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval+1:n-
pilotInterval+2)) + hopt2(1)*f(n-pilotInterval+2) + hopt2(2)*f(n-pilotInterval+1);
            r_temp = demodulate(hdemod,m(n-pilotInterval+2)/a*conj(h_upd)/abs(h_upd));
            z_temp = modulate(hmod,r_temp);
            h_temp = m(n-pilotInterval+2)/z_temp;
            if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
                r(n-pilotInterval+2) = r_temp;
                f(n-pilotInterval+2) = h_temp;
            end
            hopt = MMSE(f(n-pilotInterval+1:n), min_filter_length, Rs);
            for i = min_filter_length:pilotInterval
                h_upd =f(n-pilotInterval)-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-pilotInterval+i-
2:n-pilotInterval+i)) + hopt(1)*f(n-pilotInterval+i) + hopt(2)*f(n-pilotInterval+i-1) + hopt(3)*f(n-
pilotInterval+i-2);
                r_temp = demodulate(hdemod,m(n-pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
                z_temp = modulate(hmod,r_temp);
                h_temp = m(n-pilotInterval+i)/z_temp;
                if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
                   r(n-pilotInterval+i) = r_temp;
                   f(n-pilotInterval+i) = h_upd;
                   f(n-pilotInterval+i+1) = h_temp;
                else if n-pilotInterval+i == 3
                   h_upd = f(n-pilotInterval)-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval+i-2:n-
pilotInterval+i-1)) + hopt2(1)*f(n-pilotInterval+i-1) + hopt2(2)*f(n-pilotInterval+i-2);
                   r_temp = demodulate(hdemod,m(n-pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
                   z_temp = modulate(hmod,r_temp);
                   h_temp = m(n-pilotInterval+i)/z_temp;
                   r(n-pilotInterval+i) = r_temp;
                   f(n-pilotInterval+i+1) = h_temp;
                   f(n-pilotInterval+i) = h_upd;
                   else
                      h_upd = f(n-pilotInterval)-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-1)) + hopt(1)*f(n-pilotInterval+i-1) + hopt(2)*f(n-
pilotInterval+i-2) + hopt(3)*f(n-pilotInterval+i-3);
```

```
                    r(n-pilotInterval+i) = demodulate(hdemod,m(n-
pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
                    z_temp = modulate(hmod,r(n-pilotInterval+i));
                    h_temp = m(n-pilotInterval+i)/z_temp;
                    f(n-pilotInterval+i+1) = h_temp;
                    f(n-pilotInterval+i) = h_upd;
                  end
                end
              end
        end
```

3. Minimum Mean Square Error Filter

```
        function [output] = MMSE(channel_est, order, Rs)
        % Written by Gabriel Tham
        % NPS
        % provides the MMSE estimate of the channel tap based on past channel tap
        % estimates from channel estimation

          filter_size = 1;
          hopt = zeros(1, order);
          output = zeros(1, length(channel_est));
          channel_est = transpose(channel_est);
          channel_est_zeromean = channel_est - mean(channel_est);
          Cov = xcorr(channel_est_zeromean, 'biased');
          Cov_s = Cov((length(Cov)+1)/2:((length(Cov)+1)/2)+order-1);
          Cov_ch_est = toeplitz(Cov_s(1:order));
          for k = 2:1:order
            Cov_ch_est(k,1) = (Cov((length(Cov)+1)/2-order+1));
          end
          for i = 1:1:order-1
            filter_size = [filter_size besselj(0,2*pi*50*i/Rs)];
          end
          mean_chan = mean(channel_est);
          error = channel_est-channel_est(1);
          mean_err = mean(error);
          Cov2 = mean(abs(channel_est(1)).^2)*conj(filter_size)-((mean_chan-
mean_err)*mean_err);
          Cov2_est = Cov2(1:order);
          if Cov_ch_est == zeros(order, order)
            hopt(1) = 1;
            output = hopt;
          else
            hopt = inv(Cov_ch_est)*transpose(Cov2_est);
            output = hopt./(norm(hopt));
          end
        end
```

C. DIFFERENTIAL DECISION-FEEDBACK AND DOUBLE-DIFFERENTIAL DECISION-FEEDBACK WITH MINIMUM MEAN SQUARE ERROR AND ADAPTIVE TOLERANCE

1. Differential Decision-Feedback With MMSE and Adaptive Tolerance

```
% Written by Gabriel Tham
% NPS
% Demodulates using D-DF algorithm with MMSE and adaptive tolerance used

function r = D_DF_demod_MMSE_AT(m, hmod, hdemod, pilotSymbol, pilotInterval,
Rs, Tol_factor)
% NPS
% Demodulates using D-DF algorithm with MMSE and Adpative Tolerance

Tol = .1;
r = zeros(1,length(m));     % represents the demodulated received message
f = zeros(1,length(m));     % records all phase errors
min_filter_length = 3;
zpilot = modulate(hmod,pilotSymbol);
PS_Num = 0;
for n = 1:length(m)
   if and(PS_Num >=1, rem(n,pilotInterval) == 1)
      Tol2 = min(Tol_factor*var(f(n-pilotInterval:n-1)),Tol);
      hopt2 = MMSE(f(n-pilotInterval:n-1), min_filter_length-1, Rs);
      h_upd =(f(n-pilotInterval)+f(n))/2-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval:n-
pilotInterval+1)) + hopt2(1)*f(n-pilotInterval+1) + hopt2(2)*f(n-pilotInterval);
      r_temp = demodulate(hdemod,m(n-pilotInterval+1)/a*conj(h_upd)/abs(h_upd));
      f(n-pilotInterval+1) = h_upd;
      z_temp = modulate(hmod,r_temp);
      h_temp = m(n-pilotInterval+1)/z_temp;
      if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
         r(n-pilotInterval+1) = r_temp;
         f(n-pilotInterval+1) = h_upd;
         f(n-pilotInterval+2) = h_temp;
      end
      hopt = MMSE(f(n-pilotInterval:n-1), min_filter_length, Rs);
      for i = min_filter_length:pilotInterval
         h_upd =(f(n-pilotInterval)+f(n))/2-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-1)) + hopt(1)*f(n-pilotInterval+i-1) + hopt(2)*f(n-
pilotInterval+i-2) + hopt(3)*f(n-pilotInterval+i-3);
         r_temp = demodulate(hdemod,m(n-pilotInterval+i-1)/a*conj(h_upd)/abs(h_upd));
         z_temp = modulate(hmod,r_temp);
         h_temp = m(n-pilotInterval+i-1)/z_temp;
```

```
                if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
                    r(n-pilotInterval+i-1) = r_temp;
                    f(n-pilotInterval+i-1) = h_upd;
                    f(n-pilotInterval+i) = h_temp;
                else if n-pilotInterval+i-1 == 3
                    h_upd =(f(n-pilotInterval)+f(n))/2-(hopt2(1)+hopt2(2))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-2)) + hopt2(1)*f(n-pilotInterval+i-2) + hopt2(2)*f(n-
pilotInterval+i-3);
                    r_temp = demodulate(hdemod,m(n-pilotInterval+i-
1)/a*conj(h_upd)/abs(h_upd));
                    z_temp = modulate(hmod,r_temp);
                    h_temp = m(n-pilotInterval+i-1)/z_temp;
                    r(n-pilotInterval+i-1) = r_temp;
                    f(n-pilotInterval+i) = h_temp;
                    f(n-pilotInterval+i-1) = h_upd;
                    else
                        h_upd =(f(n-pilotInterval)+f(n))/2-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-4:n-pilotInterval+i-2)) + hopt(1)*f(n-pilotInterval+i-2) + hopt(2)*f(n-
pilotInterval+i-3) + hopt(3)*f(n-pilotInterval+i-4);
                        r_temp = demodulate(hdemod,m(n-pilotInterval+i-
1)/a*conj(h_upd)/abs(h_upd));
                        z_temp = modulate(hmod,r_temp);
                        h_temp = m(n-pilotInterval+i-1)/z_temp;
                        r(n-pilotInterval+i-1) = r_temp;
                        f(n-pilotInterval+i) = h_temp;
                        f(n-pilotInterval+i-1) = h_upd;
                    end
                end
            end
        end
        if rem(n,pilotInterval) == 1    % identify the pilot symbols
            r(n) = pilotSymbol;        % let received symbol be pilot symbol
            h = m(n) / zpilot;         % determine channel tap
            e = angle(h);           % calculate channel tap phase error
            e1 = 0;
            a = abs(h);               % calculate channel tap magnitude
            pilot_channel = h;
            f(n) = pilot_channel;
            original_chan(n) = f(n);
            PS_Num = PS_Num + 1;
        else
            r(n) = demodulate(hdemod,m(n)/a*conj(h)/abs(h)*exp(-1i*e1));    % demodulate
with channel tap info and removed a
            z = modulate(hmod,r(n));                        % calculate new channel tap
            h1 = m(n) / z;
            eh = -e+angle(h1);
            if or(abs(eh-e1)< Tol, and(and(abs(eh)>(pi-Tol),abs(e1)>(pi-Tol)),or(abs(eh-e1-
2*pi)< Tol,abs(eh-e1+2*pi)< Tol))) % updated with more accurate measure of Tol
```

```
                f(n) = h1;
                e1 = eh;
            else
                f(n) = h1;
            end
        end
    end
    if mod(n,pilotInterval)~= 0
        pilotInterval = mod(n,pilotInterval);
    end
        Tol2 = min(Tol_factor*var(f(n-pilotInterval:n-1)),Tol);
        hopt2 = MMSE(f(n-pilotInterval+1:n), min_filter_length-1, Rs);
        h_upd = f(n-pilotInterval+1)-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval+1:n-
pilotInterval+2)) + hopt2(1)*f(n-pilotInterval+2) + hopt2(2)*f(n-pilotInterval+1);
        r_temp = demodulate(hdemod,m(n-pilotInterval+2)/a*conj(h_upd)/abs(h_upd));
        f(n-pilotInterval+2) = h_upd;
        z_temp = modulate(hmod,r_temp);
        h_temp = m(n-pilotInterval+2)/z_temp;
        if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
            r(n-pilotInterval+2) = r_temp;
            f(n-pilotInterval+2) = h_upd;
            f(n-pilotInterval+3) = h_temp;
        end
        hopt = MMSE(f(n-pilotInterval+1:n), min_filter_length, Rs);
        for i = min_filter_length:pilotInterval
            h_upd =f(n-pilotInterval)-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-pilotInterval+i-
2:n-pilotInterval+i)) + hopt(1)*f(n-pilotInterval+i) + hopt(2)*f(n-pilotInterval+i-1) + hopt(3)*f(n-
pilotInterval+i-2);
            r_temp = demodulate(hdemod,m(n-pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
            z_temp = modulate(hmod,r_temp);
            h_temp = m(n-pilotInterval+i)/z_temp;
            if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
                r(n-pilotInterval+i) = r_temp;
                f(n-pilotInterval+i) = h_upd;
                f(n-pilotInterval+i+1) = h_temp;
            else if n-pilotInterval+i == 3
                h_upd = f(n-pilotInterval)-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval+i-2:n-
pilotInterval+i-1)) + hopt2(1)*f(n-pilotInterval+i-1) + hopt2(2)*f(n-pilotInterval+i-2);
                r_temp = demodulate(hdemod,m(n-pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
                z_temp = modulate(hmod,r_temp);
                h_temp = m(n-pilotInterval+i)/z_temp;
                r(n-pilotInterval+i) = r_temp;
                f(n-pilotInterval+i+1) = h_temp;
                f(n-pilotInterval+i) = h_upd;
                else
```

```
                    h_upd = f(n-pilotInterval)-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-1)) + hopt(1)*f(n-pilotInterval+i-1) + hopt(2)*f(n-
pilotInterval+i-2) + hopt(3)*f(n-pilotInterval+i-3);
                    r_temp = demodulate(hdemod,m(n-
pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
                    z_temp = modulate(hmod,r_temp);
                    h_temp = m(n-pilotInterval+i)/z_temp;
                    r(n-pilotInterval+i) = r_temp;
                    f(n-pilotInterval+i+1) = h_temp;
                    f(n-pilotInterval+i) = h_upd;
                  end
                end
              end
        end
```

2. Double-Differential Decision-Feedback With MMSE and Adaptive Tolerance

```
        % Written by Gabriel Tham
        % NPS
        % Demodulates using DD-DF algorithm with MMSE and adaptive tolerance used

        function r = DD_DF_demod_MMSE_AT(m, hmod, hdemod, pilotSymbol, pilotInterval,
Rs, Tol_factor)
        % NPS
        % Demodulates using DD-DF algorithm with MMSE

        Tol = .1;
        r = zeros(1,length(m));     % represents the demodulated received message
        f = zeros(3,length(m));     % records all phase errors
        min_filter_length = 3;
        zpilot = modulate(hmod,pilotSymbol);
        PS_Num = 0;
        for n = 1:length(m)
           if and(PS_Num >=1, rem(n,pilotInterval) == 1)
              Tol2 = min(Tol_factor*var(f(n-pilotInterval:n-1)),Tol);
              hopt2 = MMSE(f(n-pilotInterval:n-1), min_filter_length-1, Rs);
              h_upd =(f(n-pilotInterval)+f(n))/2-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval:n-
pilotInterval+1)) + hopt2(1)*f(n-pilotInterval+1) + hopt2(2)*f(n-pilotInterval);
              r_temp = demodulate(hdemod,m(n-pilotInterval+1)/a*conj(h_upd)/abs(h_upd));
              z_temp = modulate(hmod,r_temp);
              h_temp = m(n-pilotInterval+1)/z_temp;
              if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
                 r(n-pilotInterval+1) = r_temp;
                 f(n-pilotInterval+1) = h_temp;
              end
              hopt = MMSE(f(n-pilotInterval:n-1), min_filter_length, Rs);
```

```
            for i = min_filter_length:pilotInterval
                h_upd =(f(n-pilotInterval)+f(n))/2-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-1)) + hopt(1)*f(n-pilotInterval+i-1) + hopt(2)*f(n-
pilotInterval+i-2) + hopt(3)*f(n-pilotInterval+i-3);
                r_temp = demodulate(hdemod,m(n-pilotInterval+i-1)/a*conj(h_upd)/abs(h_upd));
                z_temp = modulate(hmod,r_temp);
                h_temp = m(n-pilotInterval+i-1)/z_temp;
                if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
                    r(n-pilotInterval+i-1) = r_temp;
                    f(n-pilotInterval+i) = h_temp;
                else if n-pilotInterval+i-1 == 3
                    h_upd =(f(n-pilotInterval)+f(n))/2-(hopt2(1)+hopt2(2))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-2)) + hopt2(1)*f(n-pilotInterval+i-2) + hopt2(2)*f(n-
pilotInterval+i-3);
                    r_temp = demodulate(hdemod,m(n-pilotInterval+i-
1)/a*conj(h_upd)/abs(h_upd));
                    z_temp = modulate(hmod,r_temp);
                    h_temp = m(n-pilotInterval+i-1)/z_temp;
                    r(n-pilotInterval+i-1) = r_temp;
                    f(n-pilotInterval+i) = h_temp;
                    f(n-pilotInterval+i-1) = h_upd;
                    else
                        h_upd =(f(n-pilotInterval)+f(n))/2-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-4:n-pilotInterval+i-2)) + hopt(1)*f(n-pilotInterval+i-2) + hopt(2)*f(n-
pilotInterval+i-3) + hopt(3)*f(n-pilotInterval+i-4);
                        r(n-pilotInterval+i-1) = demodulate(hdemod,m(n-pilotInterval+i-
1)/a*conj(h_upd)/abs(h_upd));
                        z_temp = modulate(hmod,r(n-pilotInterval+i-1));
                        h_temp = m(n-pilotInterval+i-1)/z_temp;
                        f(n-pilotInterval+i) = h_temp;
                        f(n-pilotInterval+i-1) = h_upd;
                    end
                end
            end
        end
        if rem(n,pilotInterval) == 1    % identify the pilot symbols
            r(n) = pilotSymbol;        % let received symbol be pilot symbol
            h = m(n) / zpilot;         % determine channel tap
            e = angle(h);           % calculate channel tap phase error
            e1 = 0;
            e2 = 0;
            a = abs(h);               % calculate channel tap magnitude
            f(n) = h;
            PS_Num = PS_Num + 1;
        else
            r(n) = demodulate(hdemod,m(n)/a*conj(h)/abs(h)*exp(-1i*(2*e1-e2)));
            z = modulate(hmod,r(n));                        % calculate new channel tap
            h1 = m(n) / z;
```

```
            eh = -e+angle(m(n)/z);
            if or(((abs(eh-e1))< Tol),and(and(abs(e1+e)>(pi-Tol),abs(eh+e)>(pi-Tol)),or(abs(eh-
e1-2*pi)< Tol,abs(eh-e1+2*pi)< Tol)))
               f(n) = h1;
               e2 = e1;
               e1 = eh;
               h=h1;
            else
               f(n) = f(n-1);
             end
          end
       end
       if mod(n,pilotInterval)~= 0
          pilotInterval = mod(n,pilotInterval);
       end
            Tol2 = min(Tol_factor*var(f(n-pilotInterval:n-1)),Tol);
            hopt2 = MMSE(f(n-pilotInterval+1:n), min_filter_length-1, Rs);
            h_upd = f(n-pilotInterval+1)-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval+1:n-
pilotInterval+2)) + hopt2(1)*f(n-pilotInterval+2) + hopt2(2)*f(n-pilotInterval+1);
            r_temp = demodulate(hdemod,m(n-pilotInterval+2)/a*conj(h_upd)/abs(h_upd));
            z_temp = modulate(hmod,r_temp);
            h_temp = m(n-pilotInterval+2)/z_temp;
            if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
               r(n-pilotInterval+2) = r_temp;
               f(n-pilotInterval+2) = h_temp;
            end
            hopt = MMSE(f(n-pilotInterval+1:n), min_filter_length, Rs);
            for i = min_filter_length:pilotInterval
               h_upd =f(n-pilotInterval)-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-pilotInterval+i-
2:n-pilotInterval+i)) + hopt(1)*f(n-pilotInterval+i) + hopt(2)*f(n-pilotInterval+i-1) + hopt(3)*f(n-
pilotInterval+i-2);
               r_temp = demodulate(hdemod,m(n-pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
               z_temp = modulate(hmod,r_temp);
               h_temp = m(n-pilotInterval+i)/z_temp;
               if or(abs(angle(h_temp)-angle(h_upd)) < Tol2, and(and(abs(angle(h_temp))>(pi-
Tol2), abs(angle(h_upd))>(pi-Tol2)), or(abs(h_temp-h_upd-2*pi)< Tol2,abs(h_temp-
h_upd+2*pi)< Tol2)))
                  r(n-pilotInterval+i) = r_temp;
                  f(n-pilotInterval+i) = h_upd;
                  f(n-pilotInterval+i+1) = h_temp;
               else if n-pilotInterval+i == 3
                  h_upd = f(n-pilotInterval)-(hopt2(1)+hopt2(2))*mean(f(n-pilotInterval+i-2:n-
pilotInterval+i-1)) + hopt2(1)*f(n-pilotInterval+i-1) + hopt2(2)*f(n-pilotInterval+i-2);
                  r_temp = demodulate(hdemod,m(n-pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
                  z_temp = modulate(hmod,r_temp);
                  h_temp = m(n-pilotInterval+i)/z_temp;
                  r(n-pilotInterval+i) = r_temp;
                  f(n-pilotInterval+i+1) = h_temp;
```

```
                    f(n-pilotInterval+i) = h_upd;
                    else
                        h_upd = f(n-pilotInterval)-(hopt(1)+hopt(2)+hopt(3))*mean(f(n-
pilotInterval+i-3:n-pilotInterval+i-1)) + hopt(1)*f(n-pilotInterval+i-1) + hopt(2)*f(n-
pilotInterval+i-2) + hopt(3)*f(n-pilotInterval+i-3);
                        r(n-pilotInterval+i) = demodulate(hdemod,m(n-
pilotInterval+i)/a*conj(h_upd)/abs(h_upd));
                        z_temp = modulate(hmod,r(n-pilotInterval+i));
                        h_temp = m(n-pilotInterval+i)/z_temp;
                        f(n-pilotInterval+i+1) = h_temp;
                        f(n-pilotInterval+i) = h_upd;
                    end
                end
            end
        end
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman, Code EC
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Tri Ha
   Naval Postgraduate School
   Monterey, California

5. Su Weilian
   Naval Postgraduate School
   Monterey, California

6. Tat Soon Yeo
   Temasek Defence Systems Institute (TDSI), National University of Singapore
   Singapore

7. Tan Lai Poh
   Temasek Defence Systems Institute (TDSI), National University of Singapore
   Singapore

8. Gabriel Tham Chi Mun
   Naval Postgraduate School
   Monterey, California