ABSTRACT

Title of Thesis: APPLICATION SPECIFIC PRECISION ANALYSIS OF CHOLESKY DECOMPOSITION IN MMSE MIMO RECEIVER SYSTEMS
Muhammad Umer Ikram, Master of Science, 2010

Directed By: Assistant Professor, Dr. Peter Petrov
Department of Electrical and Computer Engineering

We conduct an exploration study of various bit precisions for Cholesky decomposition. This research focuses on obtaining the minimum required signal to noise ratio (SNR) in Cholesky decomposition by reducing the internal precision of the computation. Primary goal of this research is to minimize resources and reduce power by performing calculations at a lower internal precision than the full 32-bit fixed or floating point. Cholesky decomposition is a key component in minimum mean square error (MMSE) multiple-input multiple-output (MIMO) receiver systems. It is used to calculate inverse of a matrix in many modern wireless systems. Cholesky decomposition is a very computation heavy process. We have investigated the effects of internal bit precisions in Cholesky decomposition. This is an exploration study to provide a benchmark for system designers to help decide on the internal precision of their system given SNR_{line}, signal and noise variances, required output SNR and symbol error rate.

Using pseudo floating point to control internal bit precision we have simulated Cholesky decomposition at various internal bit precisions with variable signal and noise variances, and SNR_{line}values. These simulations have provided SNR for lower triangular

matrix $\boldsymbol{L}$, its inverse $\boldsymbol{L}^{-1}$, and the solution vector $\boldsymbol{x}$ (from the matrix equation $\boldsymbol{Ax} = \boldsymbol{b}$). In order to observe the effects of various bit precisions on SNR and symbol error probability, SNR in $\boldsymbol{L}$ and $\boldsymbol{L}^{-1}$are plotted against condition number for 2x2, 4x4, 8x8, and 16x16 input matrices, and loss in symbol error probability (P_{sym}) is plotted against condition number for 4x4 matrices for QPSK, 16QAM and 64QAM constellations.

We find that as the internal precision is lowered there is a loss in SNR for $\boldsymbol{L}$ and $\boldsymbol{L}^{-1}$matrices. It is further observed that loss in symbol error rate is negligible for internal bit precisions of 28 bits and 24 bits in all constellations. The loss in symbol error rate begins to show at 20 bits of precision and then increases drastically, especially for higher SNR_{line}. These results provide an excellent resource for system designers. With these benchmarks, designers can decide on the internal precision of their systems according to their specifications.

APPLICATION SPECIFIC PRECISION ANALYSIS OF CHOLESKY DECOMPOSITION IN MIMO RECEIVER SYSTEMS

by

Muhammad Umer Ikram

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2010

Advisory Committee:
Assistant Professor Dr. Peter D. Petrov – Advisor
Associate Professor Emeritus Steven A. Tretter
Professor Shuvra S. Bhattacharyya

ACKNOWLEDGEMENTS

First, I would like to thank Graduate Studies Office at Electrical & Computer Engineering Department at University of Maryland College Park, and Texas Instruments in providing me the invaluable opportunity to be a Texas Instruments Scholar for the year 2008 – 2009. This opportunity enabled me to work on industry level projects and learn from experienced professionals. I would to give a special thanks to Brian Johnson, Alexander Purkovic, Mingjian Yan, and their team at TI for their patience, guidance, and support throughout my research. It was a great privilege to work with such talented people. Without their help, this research project would not have been possible.

I would like to thank my advisor, Dr. Peter Petrov, for mentoring me as a graduate student. He has always believed in me and supported me through my entire graduate studies. He will be my advisor during my Ph.D. studies as well. I would also like to thank Dr. Steven Tretter for his continual help and support throughout my research. The time he spent with me to discuss ideas and answer my questions was invaluable. I also would like to express my sincere appreciation to Dr. Shuvra Bhattacharyya for serving on my Master's Thesis committee.

Finally, I would like to acknowledge the unconditional support given to me by my mother, and father. They have always guided me in the correct direction, and helped me face challenges throughout my life. I know that I will never be able to fully repay my parents for all the hardship they have endured on my part, but at the same time I feel blessed that Allah has given me such caring parents. Also, I would like to thank my beloved wife for all her support, and for encouraging me in times of need.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1. Introduction

The current generations of mobile telecommunication networks are collectively known as 3G (“third generation"). LTE (Long Term Evolution) is the next major step towards the 4th generation (4G) of mobile radio communications designed to increase the capacity and speed of mobile networks. LTE is a set of enhancements, to the Universal Mobile Telecommunications System (UMTS), which will be introduced in 3rd Generation Partnership Project (3GPP) Release 8 [1]. LTE uses orthogonal frequency division multiplexing (OFDM) as its radio access technology, together with advanced antenna technologies. Cholesky decomposition is a key component in a 3GPP Long-Term Evolution (LTE) minimum mean square error (MMSE) multiple-input multiple-output (MIMO) receiver system. It is used to obtain the numerical solution of linear equations. The computation time in Cholesky decomposition is of the order of $n^3/3$, where n is the dimension of the matrix. Since, an MMSE MIMO receiver is implemented using digital signal processors (DSP’s), most of the DSP time and energy is spent on doing heavy matrix computations. A DSP is usually not very flexible in providing variable bit precisions to internal calculations. In order to save power, it is suggested that a co-processor be designed to perform the matrix operations at a lower bit precision than the standard 32-bits that the DSP provides. This will not only help in reducing the power cost of performing the computations but also free up the DSP for other tasks.

1.2. Cholesky Decomposition

Cholesky decomposition is used to solve a system of linear equations where the coefficient matrix is positive definite. If a matrix $\boldsymbol{A}$ is Hermitian and positive definite, then $\boldsymbol{A}$ can be decomposed as $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\dagger}$. Where $\boldsymbol{L}$ is a lower triangular matrix with strictly positive diagonal entries, and $\boldsymbol{L}^{\dagger}$ denotes the conjugate transpose of $\boldsymbol{L}$. This is defined as the Cholesky decomposition of matrix $\boldsymbol{A}$. The Cholesky decomposition is unique in the sense that there is only one lower triangular matrix $\boldsymbol{L}$ with strictly positive diagonal entries for a given Hermitian, positive-definite matrix $\boldsymbol{A}$ [3], [4].

Cholesky decomposition is used to solve the set of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ as follows:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1.2.1}$$

$\boldsymbol{L}\boldsymbol{L}^{\dagger}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\dagger}$

$$\boldsymbol{L}^{\dagger}\boldsymbol{x} = \boldsymbol{L}^{-1}\boldsymbol{b}$$

$$\boldsymbol{x} = \boldsymbol{L}^{\dagger^{-1}}\boldsymbol{L}^{-1}\boldsymbol{b} \tag{1.2.2}$$

$$\boldsymbol{x} = (\boldsymbol{L}\boldsymbol{L}^{\dagger})^{-1}\boldsymbol{b}$$

$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ where $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\dagger}$

The primary object is to solve equation (1.2.1). We use equation (1.2.2) to calculate the solution vector $\boldsymbol{x}$. Lower triangular matrix $\boldsymbol{L}$ is calculated using Cholesky decomposition. $\boldsymbol{L}^{-1}$is obtained through back-substitution. The conjugate transpose of $\boldsymbol{L}^{-1}$gives $\boldsymbol{L}^{-1^{\dagger}}$, which is the same as $\boldsymbol{L}^{\dagger^{-1}}$.

1.3. MIMO in Long Term Evolution (LTE)

Multiple-input multiple-output (MIMO) channels represent a very general description for a wide range of applications. Special cases for MIMO include Single-Input Single-Output (SISO), Multiple-Input Single-Output (MISO) and Single-Input Multiple-Output (SIMO) channels. MIMO channels are usually associated with multiple antenna systems [5]. MIMO employs multiple antennas on both the receiver and transmitter to utilize the multi-path effects that always exist to transmit additional data, thus providing increased transmission data rates without additional power. Multiple-Input Multiple-Output (MIMO) systems offer high reliability and data rate. High reliability with full diversity is achieved with the use of Orthogonal Space-Time Block Codes (OSTBC) [6], [7]. High data rates can be obtained using spatial multiplexing (SM) [8] in a MIMO system. Although MIMO adds complexity to the system in terms of processing and the number of antennas required, it enables much higher data rates along with improved spectral efficiency. As a result, MIMO has been included as an integral part of LTE. Figure 1.1 shows a general MIMO channel model.

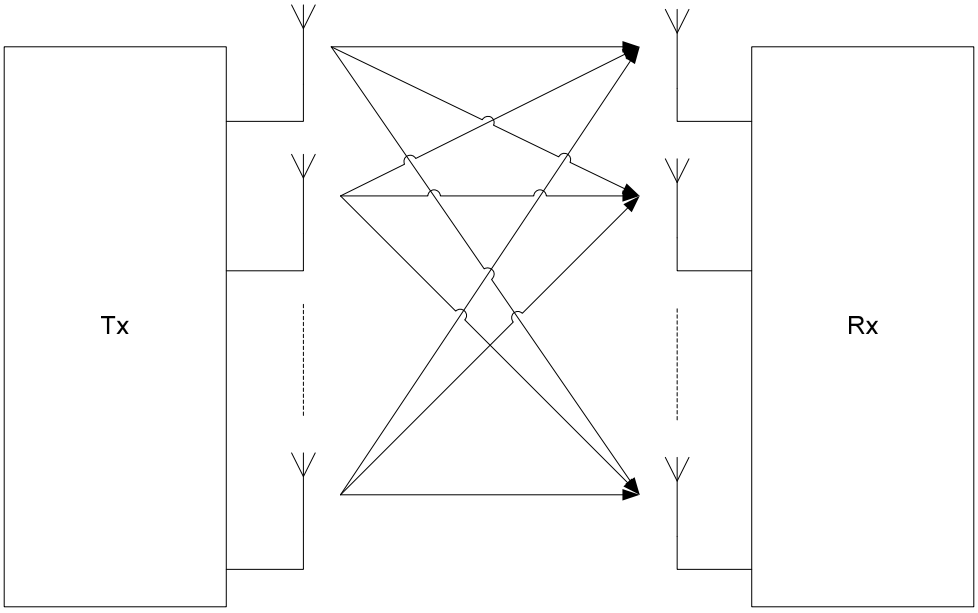

Figure 1.1: A general MIMO channel model

MIMO uses quadrature amplitude modulation (QAM) for data transmission. QAM is a modulation scheme which conveys two digital data streams by changing the amplitudes of two carrier waves, using the amplitude-shift keying (ASK) digital modulation scheme. These two carrier waves are 90° out of phase with each other and are thus called quadrature. At the receiver, the modulated waves are summed, and the resulting waveform is a combination of both phase-shift keying (PSK) and amplitude-shift keying (ASK). MIMO uses square constellations namely, quadrature phase-shift keying (QPSK, also known as 4QAM), 16QAM, and 64QAM schemes for transmission. 4, 16 and 64 indicate the number of constellation points in the QAM schemes respectively. In QAM, the constellation points are arranged in a square grid with equal vertical and horizontal spacing. Figure 1.2 and Figure 1.3 show the constellation diagrams for QPSK and 16QAM with Gray coding, where each adjacent symbol differs by only one bit. By moving to a higher-order constellation, it is possible to transmit more bits per symbol. However, if the mean energy of the constellation is to remain the same, the

points must be closer together and are thus more susceptible to noise. This results in a higher bit error rate. Therefore, higher-order QAM can deliver more data less reliably than lower-order QAM, for constant mean constellation energy. In our simulations we calculate solution vector SNR, and loss in symbol error rate for QPSK, 16QAM and 64QAM constellations, in order to cover the most commonly used transmission schemes in a MIMO system.

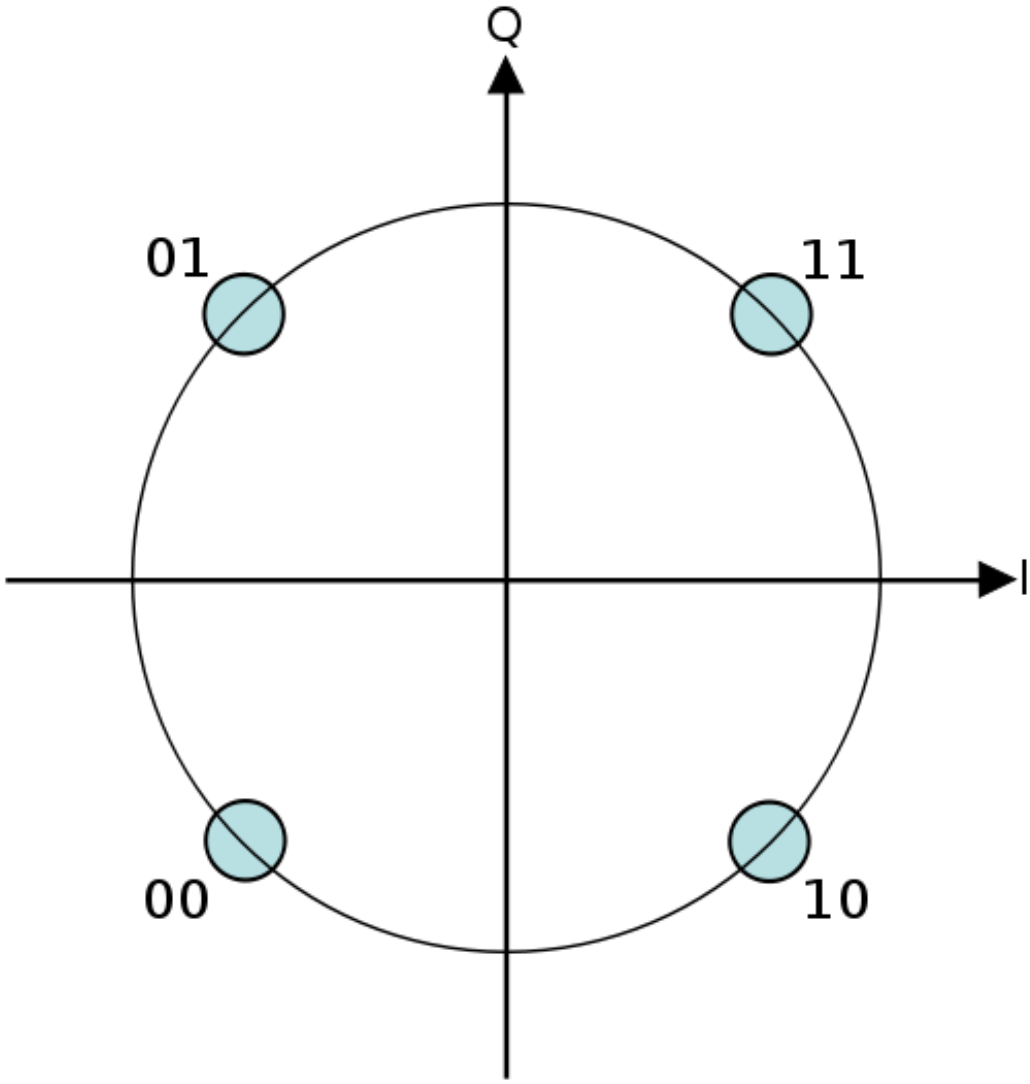

Figure 1.2: QPSK Constellation diagram with Gray coding

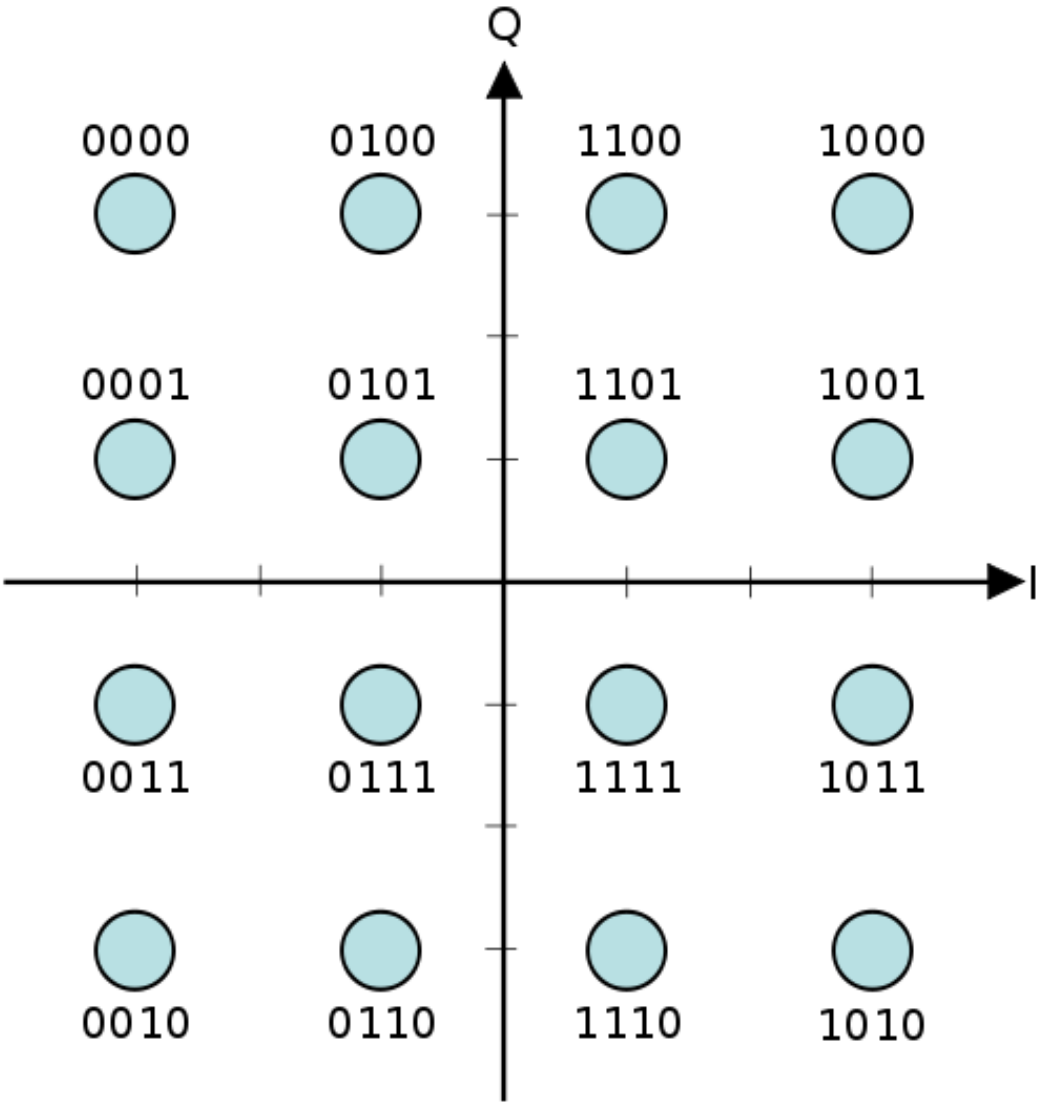

Figure 1.3: 16QAM Constellation diagram with Gray coding

1.4. Precision Analysis

We use pseudo-floating point to simulate various bit precisions in our simulations. In pseudo-floating point, a number is represented by two numerals of variable bit width, one mantissa part and one exponent part. The exponent is fixed at 8-bits. This is the minimum number of bits required to handle the range of numbers in our system. The mantissa part is varied from 10-bits to 31-bits. Our simulations do not truncate bits at the output stage, but take into account rounding errors, and overflow scenarios during each

computation step in order to provide reliable data. SNRs for the $\boldsymbol{L}$ and $\boldsymbol{L}^{-1}$matrices are calculated at 28, 32, 36, and 39-bits of precision. The SNR for solution vector $\boldsymbol{x}$, and loss in symbol error rate for QPSK, 16QAM, and 64QAM constellations are simulated at 18-bits, 20-bits, 24-bits, and 28-bits of precision. We found that there is a gradual degradation in the SNR with the decrease in bit precision. Similar degradation is observed in symbol error rate at lower bit precisions. The MMSE MIMO system designers can apply the results of our study in order to find the best bit precision for their system, given their specifications for SNR_{line}, signal and noise variances, output SNR, and loss in symbol error rate.

1.5. Outline

The thesis is arranged as follows:

- In Chapter 2, we site work done by other research groups around the world towards energy efficient MIMO systems, precision exploration studies, and matrix manipulation optimization methods.
- In Chapter 3, we provide the full problem formulation, and the motivations behind undertaking this research project.
- In Chapter 4, we describe the simulation setup, and the MIMO channel model used. This includes the choices of variables explored, and fixed and variable entities for each set of simulations are specified clearly.
- In Chapter 5, all the results are enumerated, and discussed in detail.
- In Chapter 6, we summarize the conclusions and discuss future prospects in this field of study.

Chapter 2 Related Work

Wireless communication is one of the most important Digital Signal Processing (DSP) applications. The design of low power systems is one of the key challenges in the domain of wireless communication. With an increase in the type of services provided (image, video, internet access, etc), the demand for high data rates has increased in recent years. Consequently, the complexity of the baseband digital signal is growing. Since bandwidth is limited by power within a wireless system (node or base-station), new strategies to reduce the energy and power consumption at an acceptable level must be proposed [14]. Many optimizations have been proposed for MIMO transmitters, MIMO channel, and MIMO receivers. All these efforts focus on reducing energy and power consumption in a MIMO system.

Khan et al analyzed the vertical Bell Laboratories layered space time (VBLAST) receiver used in a MIMO wireless system from a hardware implementation perspective. The primary motivation behind this is that MIMO is very expensive with regard to area and power consumption if implemented in hardware [2]. VBLAST is a MIMO detection algorithm [8] that provides a good tradeoff between bit error rate (BER) performance and computational complexity compared to its counter parts. They identify the processing elements that consume more area and power due to complex signal processing. They propose two power efficient VLSI architectures for blocks that compute pseudo-inverse of the channel matrix using SVD [16]. One is a multiplier-based architecture which consumes less power and has low area cost. This architecture uses one divider and eight multipliers with some glue logic to perform the pseudo inverse computation using the

square root algorithm [16]. The other is the improved CORDIC-based architecture that has the advantages of slightly reduced area and is able to operate at a higher frequency (160MHz) [16]. The same authors have proposed an architecture for calculating the pseudo inverse using two independent and generic pipelined CORDICs [9]. Selective clock gating is used in the MAC module to reduce the power consumption even further [16]. From a power efficiency point of view, the multiplier based implementation should be preferred.

At the University of Texas at Austin, the Wireless Networking and Communications Group has proposed an adaptive technique exploiting transmission mode switching between MIMO and SIMO. The idea is that most of the time a base station is underutilized. Due to the DC power components associated with the multiplicity of transmission chains, MIMO has higher power consumption than SIMO. Therefore, if the mode of a mobile transmitter or a base station can be switched dynamically from MIMO to SIMO according to the situation, it will be possible to save up to 50% of the mobile terminals' transmission energy [17]. Winston Ho et al propose an optimization to minimize the overall transmit power in a MIMO-OFDM downlink, if user target rates are known. This special resource allocation reduces the interference ingress to neighbouring cells and limits the power consumption at the base station [18]. The optimal solution using the proposed optimization algorithm can be found in O(KM) time for a system with K users and M subcarriers. Linear beamforming is assumed at both the transmitter and the receiver ends. To deal with the frequency-flat fading in OFDM resource allocation they propose using dual proportional fairness. This proposed

method handles all fading scenarios, including flat or partially frequency-selective fading [18].

Efficient implementation of DSP applications in embedded wireless systems requires fixed-point arithmetic. Therefore, the vast majority of embedded DSP applications are implemented in fixed-point architectures. In [14], a dynamic precision scaling method is proposed on the basis that the fixed-point specification at the MIMO receiver depends on external elements (noise level, input signal dynamic range, quality of service) and can be adapted during runtime to reduce the average power consumption. A Dynamic Precision Scaling approach that adapts the fixed-point specification according to the input receiver SNR is proposed. This enables reduction in power consumption by reducing the precision at the receiver depending on the SNR of the incoming signal. Singh et al investigated the effects of quantization noise and round off errors involved in finite-precision signal processing on the performance of MIMO receivers under flat-fading channel conditions [11]. They simulated the bit error rate (BER) performance for a range of finite precisions for two common MIMO architectures – 2 x 2 MIMO and 4 x 2 MIMO. They investigate the sources of quantization noise and round off errors in a MIMO receiver. Minimum word-length precision requirements for various MIMO schemes are also provided in [11].

Ryan Kastner and his team at University of California, San Diego are concentrating on optimizing architectures for matrix decomposition algorithms for wireless communication systems [19] [20] [21]. They have developed a new core generation tool (GUSTO) that does automatic generation and optimization of matrix decomposition methods (QR, LU, and Cholesky). GUSTO offers different parameterization options such

as resource allocation, bit widths of the data, number of functional units and organization of controllers and interconnects [19] [20] [21]. Using GUSTO, a designer can easily study the area and throughput tradeoffs of different architectures. The application specific architectures generated by GUSTO decrease the area by 83%, 94% and 86% and increase the throughput by 16%, 68%, and 14% as compared to the general purpose architecture for QR, LU, and Cholesky decompositions respectively. Currently GUSTO works on small matrix sizes (maximum 8x8) using fixed point arithmetic and architecture only [19].

Anup Hosangadi, Farzan Fallah and Ryan Kastner have also developed algebraic methods for optimizing constant multiplications in linear systems [22] [23]. This is of great importance in wireless communications. A lot of energy and power is used in solving linear equations in all wireless systems. Specifically, MIMO receiver systems employ both QR and Cholesky decompositions to calculate inverse of the received channel matrix. Commonly, constant multiplications are implemented in hardware by nesting a sequence of addition and shift operations. These can be optimized further by finding common sub-expressions among these operations. In [22] they present algebraic techniques in multi-level logic synthesis for the minimization of the number of literals and hence gates used in Boolean implementation. Authors in [22] use rectangle covering and fast extract (FX) and adapt them to the problem of optimizing linear arithmetic expressions. The main advantage of using such methods is that systems consisting of multiple variables can be optimized. These systems cannot be optimized using conventional techniques. The optimizations are aimed at reducing hardware area and power consumption [22]. Ryan Kastner's group, using experimental results, has been able

to show over 30% improvement in the number of operations over conventional techniques. Synthesis and simulation results support an equal level of area and power reduction [22].

In light of the work already done in optimizing MIMO wireless communication, we aim to explore the relationship between bit precision used in calculations and output SNR in Cholesky decomposition. Since Cholesky decomposition is a primary method in calculating the inverse of the channel matrix, optimizing the power consumed in this step will have a significant effect on the overall power of the MIMO receiver.

Chapter 3 Problem Formulation

3.1. Channel Model

A multiuser MIMO system can be categorized into a transmitter (user), and a receiver (base station). The number of transmit antennas (N_T) and the number of receive antennas (N_R) govern the sizes of all matrices and vectors in the system. The number of transmit and receive antennas in a MIMO system do not have to be the same.

3.1.1. LTE Uplink MIMO

Uplink multiple-input multiple-output (UL-MIMO) receiver at the base station (eNode B) separates signals coming from two mobile users transmitting over the same resource block (Figure 3.1). Each user transmits over a single antenna. The goal of this scheme is to enable better utilization of the channel and ultimately higher aggregate throughput.

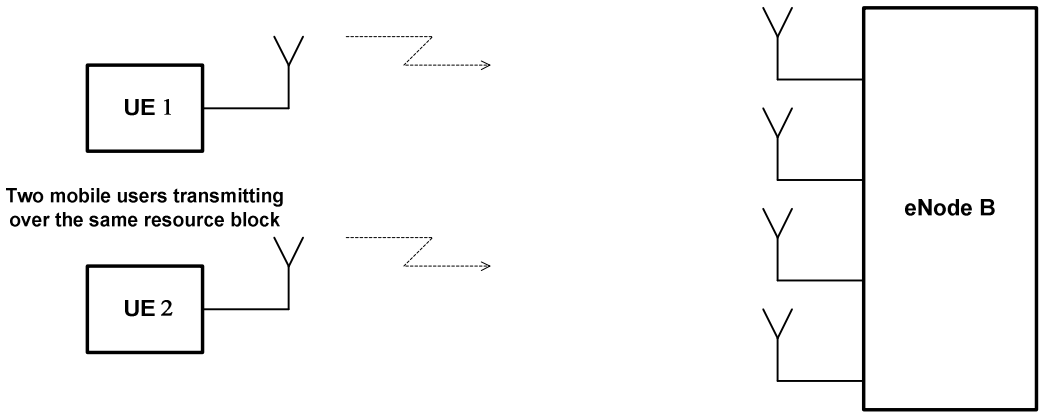

Figure 3.1: Uplink Multi-user MIMO

In addition to separating the users (layers), another function of the UL-MIMO receiver is the computation of the effective noise at the input of the constellation de-

mapper. These effective noise values are used by the soft slicer in the process of computation of the bit Log-Likelihood Ratios (bit LLR's).

3.1.2. LTE MMSE MIMO Receiver

We are interested in the MMSE MIMO receiver. It can be shown that a simplified model in the frequency domain can be used. With that assumption the system model between the output of the modulation mapper and the input to the soft slicer looks as it is shown in Figure 3.2.

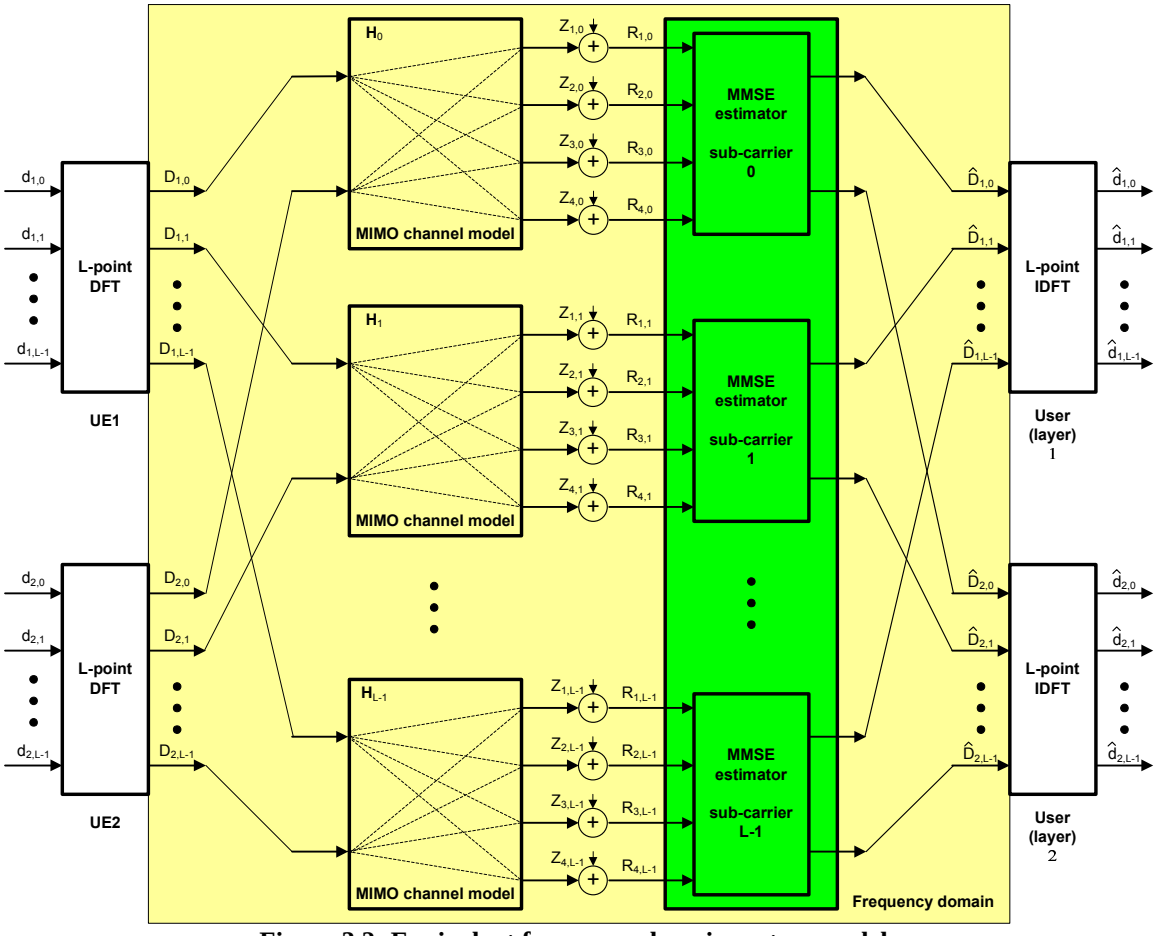

Figure 3.2: Equivalent frequency domain system model

Time domain modulated data corresponding to a particular resource block during an SC-FDMA (Single Carrier – Frequency Division Multiple Access) symbol used by UE1

and UE2 are represented by $d_{1,0}, d_{1,1}, \ldots, d_{1,L-1}$ and $d_{2,0}, d_{2,1}, \ldots, d_{2,L-1}$, respectively. Transformed data at the output of the L-point DFT (Digital Fourier Transform) block is mapped to the same set of sub-carriers in both UE's. By using this frequency domain model each sub-carrier can be processed independently.

The described frequency domain model leads to the final model that will be used for the description of the MMSE MIMO estimator. Subscripts associated with the sub-carrier index are omitted.

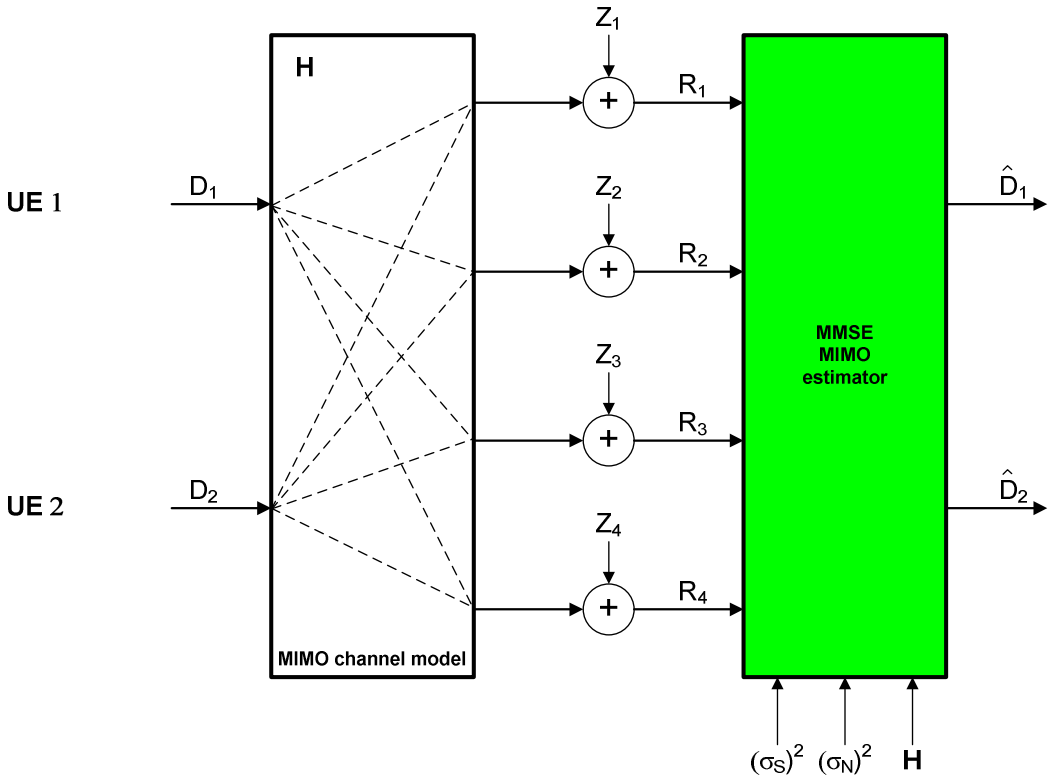

Figure 3.3: System model used for the MMSE estimation (per sub-carrier)

Based on Figure 3.3 the basic equation is:

$$\boldsymbol{R = HD + Z}$$

where:

$$\mathbf{D}=\begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \qquad \mathbf{H}=\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \\ . & \\ . & \\ H_{N_R1} & H_{N_R2} \end{bmatrix} \qquad \mathbf{R}=\begin{bmatrix} R_1 \\ R_2 \\ . \\ . \\ R_{N_R} \end{bmatrix} \qquad \mathbf{Z}=\begin{bmatrix} Z_1 \\ Z_2 \\ . \\ . \\ Z_{N_R} \end{bmatrix}$$

N_R is the number of receive antennas (2 or 4). However, the number of transmit antennas (layers, participating UE's), N_T, is fixed to 2.

The MMSE solution filter, $\boldsymbol{G}$ (an $N_T \times N_R$ matrix) is the one that minimizes:

$$J(\boldsymbol{D}) = E\{\|\boldsymbol{D} - \boldsymbol{GR}\|^2\}$$

The expectation operation is done with respect to the transmitted vector $\boldsymbol{D}$ and the noise vector $\boldsymbol{Z}$. With the assumption that signal and noise are uncorrelated, the solution filter coefficients can be found from:

$$E\{\boldsymbol{DD}^\dagger\boldsymbol{H}^\dagger - \boldsymbol{GHDD}^\dagger\boldsymbol{H}^\dagger - \boldsymbol{GZZ}^\dagger\} = \mathbf{0}_{N_T \times N_R}$$

Superscript † denotes conjugate transpose matrix. Now we introduce another reasonable assumption: The data frequency domain symbols, D_1 and D_2, transmitted by UE1 and UE2, respectively, are uncorrelated, zero-mean and their variances are the same, σ_S^2.The last part of the assumption is based on another reasonable assumption that the time domain constellation points are scaled such that their variance is fixed and independent of the modulation type. It is also assumed that differences in the transmit chains of the two participating UE's are absorbed by the channel model. With all these assumptions we have:

$$\boldsymbol{R}_{DD} = E\{\boldsymbol{DD}^\dagger\} = \sigma_S^2\boldsymbol{I}_{N_T}$$

As mentioned before the equation used for deriving the MMSE filter coefficients was:

$$E\{\boldsymbol{D}\boldsymbol{D}^{\dagger}\boldsymbol{H}^{\dagger} - \boldsymbol{G}\boldsymbol{H}\boldsymbol{D}\boldsymbol{D}^{\dagger}\boldsymbol{H}^{\dagger} - \boldsymbol{G}\boldsymbol{Z}\boldsymbol{Z}^{\dagger}\} = \boldsymbol{0}_{N_T \times N_R}$$

This equation now becomes:

$$\sigma_S^2 \boldsymbol{I}_{N_T}\boldsymbol{H}^{\dagger} - \boldsymbol{G}\boldsymbol{H}\sigma_S^2 \boldsymbol{I}_{N_T}\boldsymbol{H}^{\dagger} - \boldsymbol{G}\boldsymbol{R}_{ZZ} = \boldsymbol{0}_{N_T \times N_R}$$

$$\Rightarrow \sigma_S^2 \boldsymbol{G}\boldsymbol{H}\boldsymbol{H}^{\dagger} + \boldsymbol{G}\boldsymbol{R}_{ZZ} = \sigma_S^2 \boldsymbol{H}^{\dagger}$$

Finally, the filter coefficient matrix is computed as:

$$\boldsymbol{G} = \boldsymbol{H}^{\dagger}\left(\boldsymbol{H}\boldsymbol{H}^{\dagger} + \frac{1}{\sigma_S^2}\boldsymbol{R}_{ZZ}\right)^{-1}$$

Considering that the number of the received antennas is greater than or equal to the number of transmit antennas ($N_R \geq N_T$) the above expression can be further simplified (applying the Matrix Inversion Lemma) to:

$$\boldsymbol{G} = \left(\boldsymbol{H}^{\dagger}\boldsymbol{R}_{ZZ}^{-1}\boldsymbol{H} + \frac{1}{\sigma_S^2}\boldsymbol{I}_{N_T}\right)^{-1}\boldsymbol{H}^{\dagger}\boldsymbol{R}_{ZZ}^{-1}$$

The symbols demodulated in the frequency domain ($\widehat{\boldsymbol{D}}_{MMSE}, N_T \times 1$) are the MMSE estimate of the frequency domain samples given as [11], [12]:

$$\widehat{\boldsymbol{D}}_{MMSE} = \boldsymbol{G}\boldsymbol{R} = \left(\boldsymbol{H}^{\dagger}\boldsymbol{R}_{ZZ}^{-1}\boldsymbol{H} + \frac{1}{\sigma_S^2}\boldsymbol{I}_{N_T}\right)^{-1}\boldsymbol{H}^{\dagger}\boldsymbol{R}_{ZZ}^{-1}\boldsymbol{R} \qquad (3.1.1)$$

where $\boldsymbol{H}$, $N_R \times N_T$ is the channel matrix, $\boldsymbol{R}_{ZZ}$, $N_R \times N_R$ is the noise covariance matrix, $\boldsymbol{R}$, $N_R \times 1$ is the vector of received complex samples and σ_S^2 is the signal variance. N_T is the

number of transmit antennas and N_R is the number of receiver antennas. Transmitted symbols (*D* 's) belong to a QAM modulation constellation: QPSK, 16-QAM, 64-QAM.

We are going to focus on a special case where we assume same noise variance across all receive antennas $\rightarrow \boldsymbol{R}_{ZZ} = \sigma_N^2 \boldsymbol{I}_{N_R}$. This assumption reduces equation 3.1.1 to:

$$\widehat{\boldsymbol{D}}_{MMSE} = \boldsymbol{G}\boldsymbol{R} = \left(\boldsymbol{H}^{\dagger}\boldsymbol{H} + \frac{\sigma_N^2}{\sigma_S^2}\boldsymbol{I}_{N_T}\right)^{-1}\boldsymbol{H}^{\dagger}\boldsymbol{R}$$

Where $\boldsymbol{G} = \left(\boldsymbol{H}^{\dagger}\boldsymbol{H} + \frac{\sigma_N^2}{\sigma_S^2}\boldsymbol{I}_{N_T}\right)^{-1}\boldsymbol{H}^{\dagger}$ is the MMSE MIMO solution filter.

The minimum mean square error (covariance matrix of the effective noise in the frequency domain) of the estimator is given by:

$$\boldsymbol{R}_{VV} = cov(\boldsymbol{V})$$

$$\boldsymbol{R}_{VV} = \left(\boldsymbol{H}^{\dagger}\boldsymbol{R}_{ZZ}^{-1}\boldsymbol{H} + \frac{1}{\sigma_S^2}\boldsymbol{I}_{N_T}\right)^{-1}$$

$$\boldsymbol{R}_{VV} = \sigma_N^2\left(\boldsymbol{H}^{\dagger}\boldsymbol{H} + \frac{\sigma_N^2}{\sigma_S^2}\boldsymbol{I}_{N_T}\right)^{-1}$$

The goal is to find out how the effective noise variance changes as a function of the condition number and selected precision. Ultimately, we will show the degradation of the symbol error rate (we call D's symbols) as a function of the condition number and selected precision. One way how to assess the impact of the condition number and selected precision on the symbol error rate is to define two parameters: SNR_{line} and

signal variance σ_S^2 (noise variance is determined as: $\sigma_N^2 = \sigma_S^2/SNR_{line}$). Define $\boldsymbol{H}_e = \sigma_S \boldsymbol{H}$ and write the solution in the form:

$$\widehat{\boldsymbol{D}}_{MMSE} = \left(\boldsymbol{H}^{\dagger}\boldsymbol{H} + \frac{\sigma_N^2}{\sigma_S^2}\boldsymbol{I}_{N_T}\right)^{-1} \boldsymbol{H}^{\dagger}\boldsymbol{R} = \left(\boldsymbol{H}_e^{\dagger}\boldsymbol{H}_e + \sigma_N^2\boldsymbol{I}_{N_T}\right)^{-1}\sigma_S\boldsymbol{H}_e^{\dagger}\boldsymbol{R}$$

$\widehat{\boldsymbol{D}}_{MMSE} = \boldsymbol{A}^{-1}\boldsymbol{b}$ with $\boldsymbol{A} = \boldsymbol{H}_e^{\dagger}\boldsymbol{H}_e + \sigma_N^2\boldsymbol{I}_{N_T}$ and $\boldsymbol{b} = \sigma_S\boldsymbol{H}_e^{\dagger}\boldsymbol{R}$

σ_S^2 and σ_N^2 affect the condition number range of the matrix $\boldsymbol{A} = \boldsymbol{H}_e^{\dagger}\boldsymbol{H}_e + \sigma_N^2\boldsymbol{I}_{N_T}$. Recommendation is to use the following parameter combinations, as the lower and upper bounds for SNR_{line} are taken from industry standards in practice today. All simulations performed during this research project explore all possible combinations of σ_S^2 and SNR_{line} given in Table 3.1 for each QAM constellation.

Table 3.1: Parameter combinations for simulations

σ_S^2	QPSK, SNR_{line}(dB)		16QAM, SNR_{line}(dB)		64QAM, SNR_{line}(dB)	
2^{22} = 4194304	12	50	18	50	24	50
2^{24} = 16777216	12	50	18	50	24	50

3.2. Fixed Point Implementation Drawbacks

In this research project we investigate the new Texas Instruments (TI) C64x line of DSPs. A TI C64x DSP performs all computations in 32-bit fixed point. This provides a very high SNR for the output. In most applications such high SNR is not required. Since, the DSP is incapable of running at lower bit precisions, we suggest that a co-processor be designed that works in parallel with the DSP and performs the matrix computations at a lower bit precision. This co-processor will provide a lower but acceptable SNR at the

output. Since, the co-processor is running at a lower bit precision, it will have less complex circuitry. As a result, it will consume less power, and require fewer computational cycles as compared to the DSP. Another benefit of introducing a lower precision co-processor comes from the fact that not all the input data at the MMSE MIMO receiver is at 32-bits of precision. Most of the data is at much lower bit precision. Therefore, spending power to do 32-bit computation on a data that is inherently low rate is counter-productive. It will be beneficial to reduce power by reducing the internal bit precision of the system performing calculations on this input data (in this case, the co-processor). With these motivations, we focus our research on exploring lower bit precisions and the corresponding output SNRs.

Chapter 4 Simulation Setup

4.1. Simulation outline

First $\boldsymbol{H}_e$ matrix is generated using a custom Gaussian random number generator. All entries of $\boldsymbol{H}_e$ are complex, with real and imaginary parts of each entry at 16-bit fixed point precision. Then we take conjugate transpose of $\boldsymbol{H}_e$ to get $\boldsymbol{H}_e^{\dagger}$. The matrix product $\boldsymbol{H}_e^{\dagger}\boldsymbol{H}_e$ is calculated and then added to $\sigma_N^2\boldsymbol{I}_{N_T}$, where $\sigma_N^2 = \sigma_S^2/SNR_{line}$. σ_S^2 is the signal variance, and σ_N^2 is the noise variance in the system. This gives us the input matrix $\boldsymbol{A} = \boldsymbol{H}_e^{\dagger}\boldsymbol{H}_e + \sigma_N^2\boldsymbol{I}_{N_T}$ where each complex number entry in matrix $\boldsymbol{A}$ has 32-bits for real and imaginary parts. The resulting matrix is then converted to floating point, double floating point or custom pseudo floating point precision by calling respective conversion routines.

Using the Cholesky decomposition scheme given below we calculate $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\dagger}$. Using simple back-substitution $\boldsymbol{L}^{-1}$is computed. In order to calculate SNR for both $\boldsymbol{L}$ and $\boldsymbol{L}^{-1}$ matrices we use Matlab as a reference. We use Matlab's built-in functions to calculate $\boldsymbol{L}_{Matlab}$ and $\boldsymbol{L}_{Matlab}^{-1}$ and treat these as ideal cases. SNR (in dB) for both $\boldsymbol{L}$ and $\boldsymbol{L}^{-1}$ matrices is then calculated as follows:

$$\boldsymbol{E}_L = \boldsymbol{L}_{Matlab} - \boldsymbol{L}$$

$$\|\boldsymbol{E}_L\|^2 = \sum_{i,j} \left\|e_{ij}\right\|^2$$

$$\|\boldsymbol{L}_{Matlab}\|^2 = \sum_{i,j} \left\|\boldsymbol{l}_{Matlab_{ij}}\right\|^2$$

$$SNR_L = 10\log_{10}\left(\frac{\|\boldsymbol{L}_{Matlab}\|^2}{\|\boldsymbol{E}_L\|^2}\right)$$

Similarly for $\boldsymbol{L}^{-1}$matrix we do:

$$\boldsymbol{E}_{\boldsymbol{L}^{-1}} = \boldsymbol{L}^{-1}_{Matlab} - \boldsymbol{L}^{-1}$$

$$\|\boldsymbol{E}_{\boldsymbol{L}^{-1}}\|^2 = \sum_{i,j}\left\|e_{ij}\right\|^2$$

$$\|\boldsymbol{L}^{-1}_{Matlab}\|^2 = \sum_{i,j}\left\|\boldsymbol{l}^{-1}_{Matlab_{ij}}\right\|^2$$

$$SNR_{\boldsymbol{L}^{-1}} = 10\log_{10}\left(\frac{\|\boldsymbol{L}^{-1}_{Matlab}\|^2}{\|\boldsymbol{E}_{\boldsymbol{L}^{-1}}\|^2}\right)$$

Table 4.1: Cholesky decomposition scheme used to calculate A = LL†

```
Int32 Cholesky(
               INOUT          cplxf_t   **  matrixIn,
               INOUT          float     *   diag,
               IN             int32     n)
{
   int32   row, col, k ;
   cplxf_t cholFactor ; /* cplxf_t is a structure to represent a floating point complex number */
   Int32   cholFailed = 0 ;

   /* We are computing the lower triangular Cholesky Factor "L" col-by-col */
   for (row = 0; row < n; row++)
   {
      for (col = row; col < n; col++)
      {
          /* Assign the Cholesky Factor equal to the input matrix element. */
          cholFactor = matrixIn[row][col] ;

          /* Subtract the respective terms from the input marix element.
           * This way only the product of Cholesky Factor withh one other
           * term is left in the end. This computation is needed regardless
           * of the values of row and col variables.
           */
          for (k = 0; k < row; k++)
          {
             cholFactor = cplxSub(cholFactor, cplxMul(matrixIn[row][k], cplxConj(matrixIn[col][k])) );
          }

          /* Now we decide what to do with the Cholesky Factor based on
           * values of row and col variables.
           */
          if (row == col)
          {
             /* Input matrix, with rounding errors, is not positive definite. */
             if (cholFactor.real <= 0.0)
             {
                printf("Cholesky Decomposition failed.\n") ;
                cholFailed = 1 ;
             }

             else
             {
                /* We do a simple square root operation here because we know that the diagonal elements are all real */
                diag[row] = (float)(sqrt(cholFactor.real)) ;
             }
          }
          else
          {
             if(cholFailed == 0)
             {
                matrixIn[col][row] = cplxScalarMul(cplxConj(cholFactor), (1/diag[row])) ;
             }
          }
      } /* end of col loop */
   } /* end of row loop */
```

```
   /* Zero out the upper triangular part of the input matrix */

   for (row = 0; row < n; ++row)
      for (col = row + 1; col < n; ++col)
      {
          matrixIn[row][col].real = 0.0 ;
          matrixIn[row][col].imag = 0.0 ;
      }

   return cholFailed ;
}
```

For each realization of $\boldsymbol{H}_e$ we create M number of realizations of $\boldsymbol{R}_m$ such that $\boldsymbol{R}_m = \boldsymbol{H}\boldsymbol{D}_m$ for $m = 1,2,\ldots,M$, where $\boldsymbol{D}_m$ is a $(N_T \times \mathbf{1})$ vector of random QAM constellation points of signal variance σ_S^2. Further, for each realization of $\boldsymbol{R}_m$ we compute:

$D_m = \left\|\widehat{\boldsymbol{D}}_{MMSE\_ideal,m}\right\|^2$ and

$E_m = \left\|\widehat{\boldsymbol{D}}_{MMSE\_ideal,m} - \widehat{\boldsymbol{D}}_{MMSE,m}\right\|^2$

$\widehat{\boldsymbol{D}}_{MMSE\_ideal,m}$ is obtained using Matlab, whereas $\widehat{\boldsymbol{D}}_{MMSE,m}$ is obtained using our matrix inversion using Cholesky decomposition as described in section 1.2.

For each $\boldsymbol{H}_e$ we record condition number k of the matrix $\boldsymbol{A}$ as well as:

$$D = \frac{\sum_{m=1}^{M} \mathbf{D}_m}{MN_T} \tag{4.1.1}$$

$$E = \frac{\sum_{m=1}^{M} \mathbf{E}_m}{MN_T} \tag{4.1.2}$$

and

$$SNR_{MMSE} = D/E \tag{4.1.3}$$

Using Matlab we find linear fits to the above set of points and get $\overline{D}(k)$ and $\overline{E}(k)$ after the simulations (Section 4.4).

Figure 4.1, Figure 4.2, and Figure 4.3 show the standard plots for symbol error probability (rate) and bit error probability (rate) as a function of signal to noise ratio (SNR) for QPSK, 16QAM, and 64QAM constellations, respectively. To evaluate the loss in symbol error probability for a particular modulation at a particular precision, we find the SNR point corresponding to a given P_{sym} (say $P_{sym} = 10^{-1}$). Then we compute corresponding noise variance $\bar{N}(k) = \bar{D}(k)/SNR$. New SNR' corresponding to new conditions "spoiled" by the finite processing precision is computed as:

$$SNR' = \bar{D}(k)/\left(\bar{N}(k) + \bar{E}(k)\right) \tag{4.1.4}$$

Using SNR' and $P_{sym} = f(SNR)$ plot, or by using the equation $P'_{sym} = f(SNR')$ we calculate the new symbol error probability. Finally, the plot P_{sym}/P'_{sym} as a function of the condition number *k* provides the loss in symbol error probability.

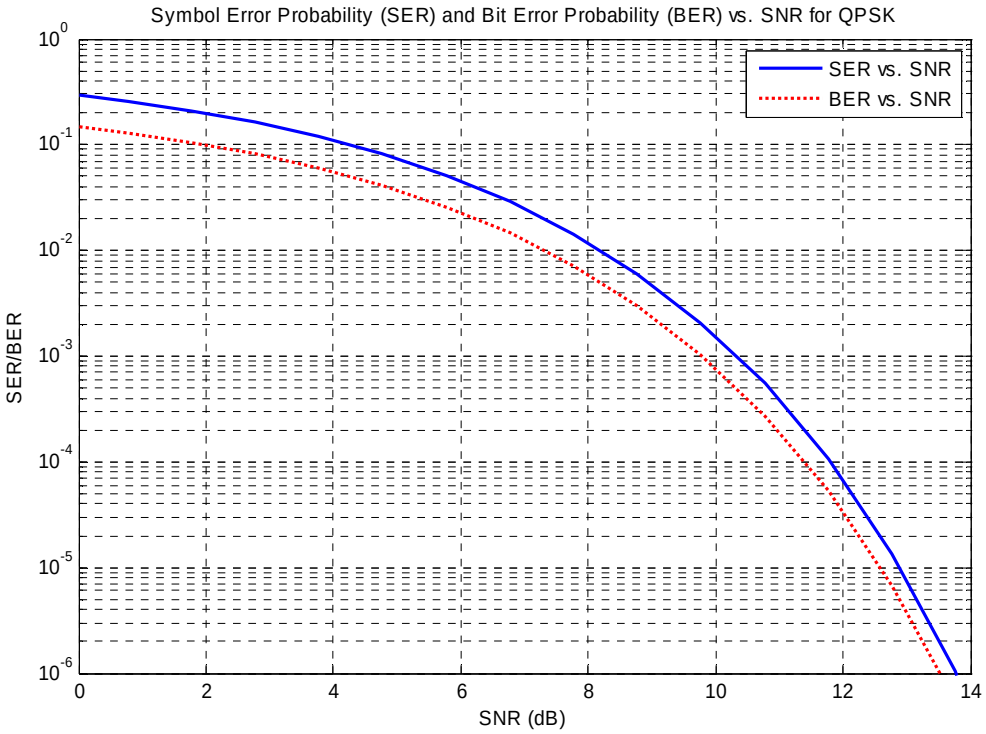

Figure 4.1: SER and BER vs. SNR for QPSK

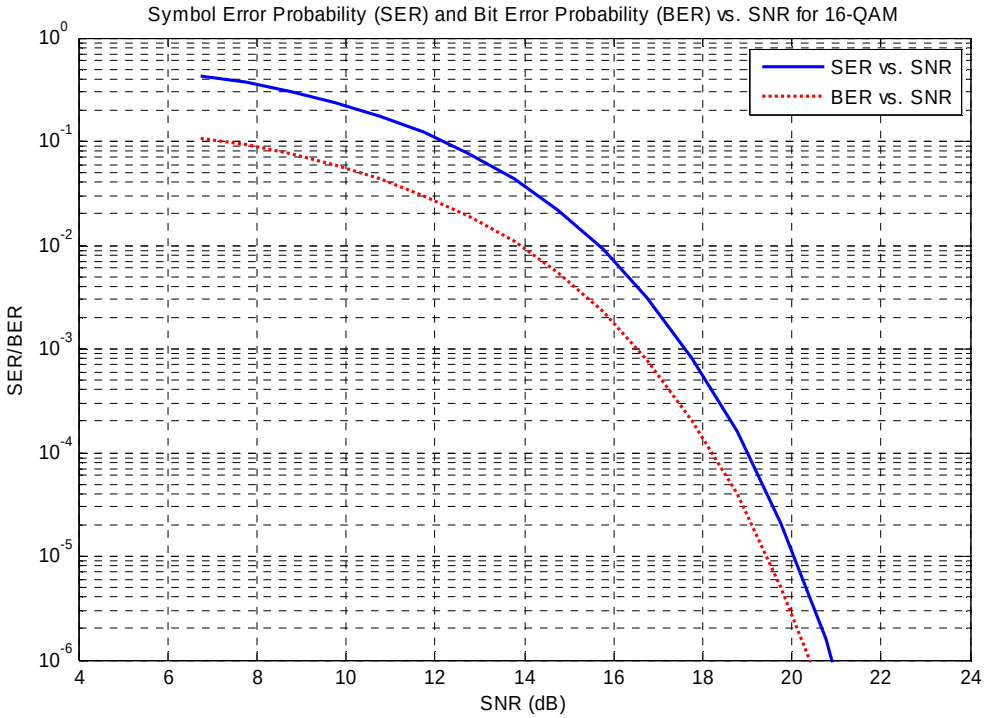

Figure 4.2: Symbol SER and BER vs. SNR for 16QAM

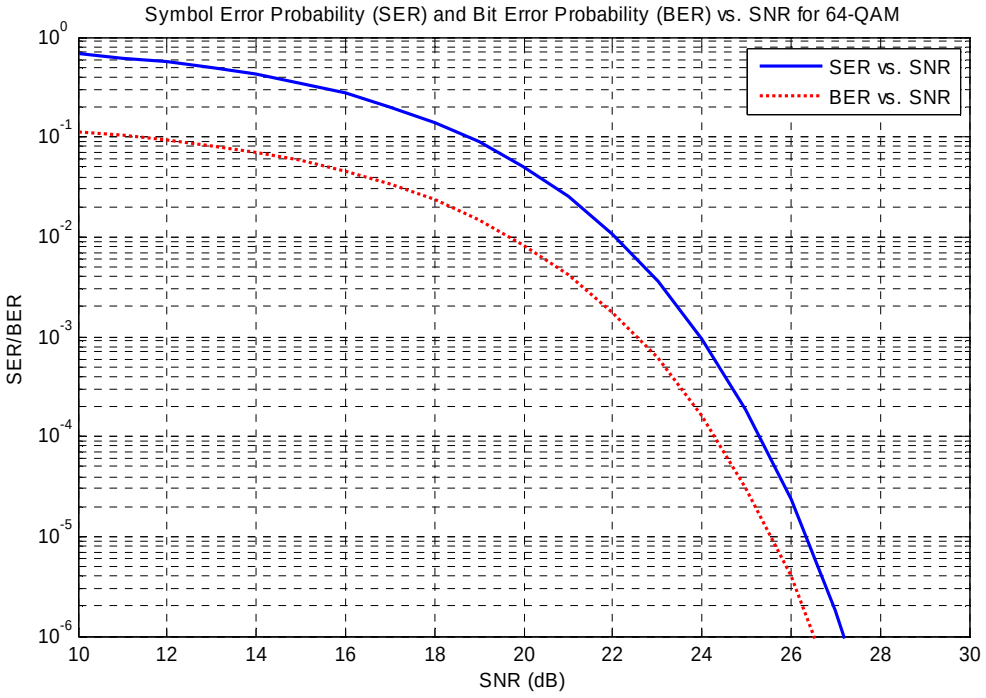

Figure 4.3: SER and BER vs. SNR for 64QAM

This process can be repeated for $P_{sym} = 10^{-2}$, $P_{sym} = 10^{-3}$, $P_{sym} = 10^{-4}$, and $P_{sym} = 10^{-5}$ depending on the system specifications. In this thesis we show plots for $P_{sym} = 10^{-1}$. Alternatively SNR loss ($SNR - SNR'$) at different P_{sym}values can also be plotted, as a measure of the signal degradation that occurs with lower precisions. We do not show SNR loss plots in this thesis.

4.2. Pseudo Floating Point

In pseudo floating point a number is represented by two integers (positive or negative), namely $Mantissa$ and $Exponent$. The number is then written as $Mantissa * 2^{Exponent}$. The precision of the number is controlled by allocating variable bits to both $Mantissa$ and $Exponent$. Precision mode is either written as $Mantissa, Exponent$, or as a single number which is the sum of $Mantissa$ and $Exponent$. During all our simulations the $Exponent$ is fixed at 8 bits. This is the minimum number of bits required to handle the range of numbers in our system. The primary motivation for using pseudo floating point is to maintain consistent internal precision throughout all calculations. At no point in the simulations we use more bits than allocated by the pseudo floating point representation. There is no truncation of bits, and all rounding errors are applied after each calculation. This is to ensure, that we simulate a system that has registers of a fixed bit-width, and is incapable of holding a number with higher bit precision. In this way unnecessary rounding-off errors are avoided as well. The advantage of using pseudo floating point is that we do not waste computation power by performing calculations at maximum precision, and then providing the answer at a lower precision by simply truncating the last few bits. By using the pseudo floating point

arithmetic we guarantee that all values remain within the operating range and overflow does not occur.

4.2.1. Precisions explored

Simulations to obtain SNR plots for $\boldsymbol{L}$ and $\boldsymbol{L}^{-1}$matrices are performed at 39-bits, 36-bits, 32-bits and 28-bits of precision. Simulations to obtain SNR plots for the solution vector $\widehat{\boldsymbol{D}}_{MMSE}$, SNR', and the ratio P_{sym}/P'_{sym}, are carried out at 28-bits, 24-bits, 20-bits, and 18-bits of precision.

4.3. SNR as a function of Condition Number

Condition number of the matrix $\boldsymbol{A}$ in linear equation (1.2.1) is a measure of the sensitivity of the system to perturbations in the transformation vector $\boldsymbol{b}$. In other words, it is a measure of error introduced in the solution vector $\boldsymbol{x}$, for any errors present in $\boldsymbol{b}$ [4]. If the co-efficient matrix $\boldsymbol{A}$ has a low condition number, then it is said to be well-conditioned. Similarly, if the condition number is high, then the matrix is said to be ill-conditioned. The condition number is a property of the matrix, and not a measure of the processing system's accuracy, nor is it a measure of algorithm efficiency. Numerically we define condition number k as [3],[4]:

$$k(\boldsymbol{A}) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$$

Condition number is also a measure of how close the matrix is to singularity. Roughly it is the difference between the highest and lowest Eigen values of the matrix. A high condition number means the matrix is close to a singular matrix, and vice versa.

Since, the condition number does not depend on numerical precision, it is a true independent variable. Therefore, we plot simulation results for SNR, SNR', and P_{sym}/P'_{sym} as a function of condition number of the coefficient matrix $\boldsymbol{A}$ in Chapter 5. This gives us a means to evaluate each simulation independently, and compare results across multiple bit precisions.

4.3.1. Effect of SNR_{line} on Condition Number

As described in Section 4.1, we define $\sigma_N^2 = \sigma_S^2/SNR_{line}$, where σ_N^2 is the noise variance, σ_S^2 is the signal variance. Since, the input matrix $\boldsymbol{A}$ is given as: $\boldsymbol{A} = \boldsymbol{H}_e^{\dagger}\boldsymbol{H}_e + \sigma_N^2\boldsymbol{I}_{N_T}$, we see that SNR_{line} has a direct impact on the condition number of $\boldsymbol{A}$. Thus, we can control the upper bound on the condition number of matrix $\boldsymbol{A}$ by changing SNR_{line}. Lower SNR_{line} upper bounds the condition number of $\boldsymbol{A}$ to a lower value, whereas a higher SNR_{line} allows higher condition numbers for matrix $\boldsymbol{A}$. Table 3.1 lists the range of SNR_{line}values most commonly used in the industry for each constellation. These values of SNR_{line} enable us to explore variable ranges of condition numbers in different constellations.

4.4. System Variables explored

The following system variables are set before each simulation:

- Number of trials
- Number of transmission antennas (N_T)
- Number of receive antennas (N_R)
- Constellation for signal modulation Simulation outline(QPSK, 16QAM, or 64QAM)

- Number of realizations of $\boldsymbol{D}_m$ for each trial, or the value of M (Section Simulation outline4.1)
- Signal variance (σ_S^2)
- SNR_{line} (in dB)
- Noise variance is calculated as $\sigma_N^2 = \sigma_S^2 / SNR_{line}$
- Values of $Integer$ and $Exponent$ to set the internal bit precision of the simulation

The following variables are measured during the simulation (Section 4.1):

- Condition number k for the matrix $\boldsymbol{A}$
- SNR (in dB) for $\boldsymbol{L}$ and $\boldsymbol{L^{-1}}$ matrices using floating point
- SNR (in dB) for $\boldsymbol{L}$ and $\boldsymbol{L^{-1}}$ matrices using pseudo floating point
- D_m, and $E_m$for each $\widehat{\boldsymbol{D}}_{MMSE,m}$
- Values of D and E (Equations 3.1.1, and 3.1.2)
- SNR_{MMSE} (in dB) (Equation 3.1.3)

The following variables are calculated after the simulations using Matlab

- $\overline{D}(k)$, $\overline{E}(k)$ and $\overline{N}(k)$
- SNR' (in dB) corresponding to each condition number
- P_{sym}/P'_{sym} corresponding to each SNR'

Chapter 5 Results & Discussion

5.1. SNR forL, L^{-1} Matrices

In this section we discuss the SNR results obtained after computing $\boldsymbol{L}$, and $\boldsymbol{L^{-1}}$through Cholesky decomposition. These results are independent of modulating constellation, as they only relate to the input coefficient matrix $\boldsymbol{A}$. We are interested in bit precisions that give an SNR greater than or equal to 50dB.

5.1.1. Simulation Specifications

For all simulation results shown in Section Chapter 0 the type of modulating constellation and the value of M are irrelevant. Following variables are constant across all simulation results:

- Number of trials is set to 5000
- Signal variance (σ_S^2) = 2^{24}
- $SNR_{line} = 50dB$
- $Exponent = 8\ bits$

Variables varied across simulation results shown in Section Chapter 0 are given below:

- $Integer$ is given the values: 31, 28, 24, and 20
- Number of transmission antennas (N_T) and the number of receive antennas (N_R) are set to give input coefficient matrices with dimensions 2×2, 4×4, 8×8, and 16×16

The figures in Section Chapter 0 also show SNR plots obtained using 32-bit floating point and the actual TI C64x chip operating at 32-bit fixed point precision for $\boldsymbol{L}$ matrices.

5.1.2. 2x2 Input Matrix

Figure 5.1 shows the SNR in dB for 2×2 $\boldsymbol{L}$ matrix as a function of condition number (k) for C64x 32-bit fixed point, 32-bit floating point, 39-bit pseudo floating point, 36-bit pseudo floating point, 32-bit pseudo floating point, and 28-bit pseudo floating point implementations. Whereas Figure 5.2 shows the SNR in dB for the corresponding $\boldsymbol{L}^{-1}$matrices. Following important observations are made from these two plots. The minimum condition number for the input coefficient matrix $\boldsymbol{A}$ is very close to 1. This is because of the small dimensions of $\boldsymbol{A}$.

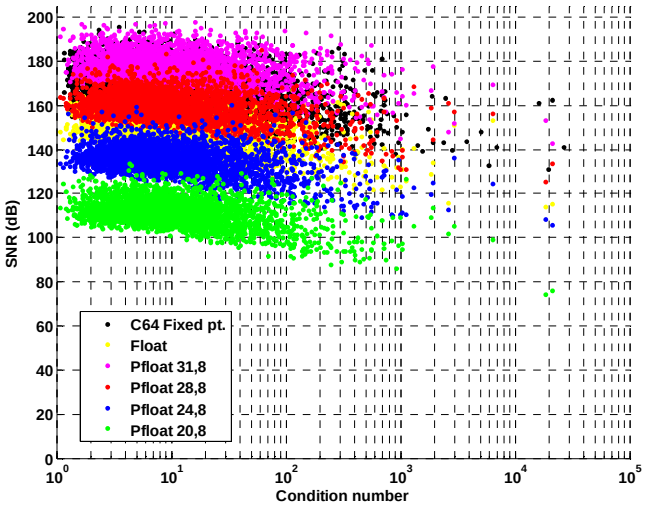

Figure 5.1: SNR(dB) for 2x2 L matrix

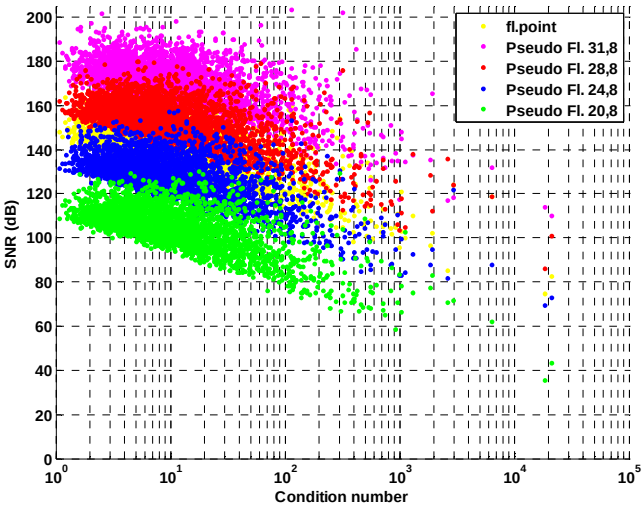

Figure 5.2: SNR(dB) for 2x2 L^{-1} matrix

The worst condition number for $\boldsymbol{A}$ is observed to be around $3\text{x}10^4$, but the bulk of the condition numbers are in the range 1 to 10^2. The 32-bit fixed point provides much higher SNR than the floating point implementation. The 32-bit fixed point SNR is very close to 39-bit pseudo floating point, and can be chosen without compromising accuracy.

Furthermore, 39-bit and 36-bit pseudo floating point implementations also give higher SNR than 32-bit floating point. However, SNRs for 32-bit and 28-bit pseudo floating point implementations are below 32-bit floating point. Moreover, the general slope of the SNRs for $\boldsymbol{L}^{-1}$ matrices is greater than that of $\boldsymbol{L}$ matrices. This is consistent with the fact that more computation is required to obtain the $\boldsymbol{L}^{-1}$ matrix than the $\boldsymbol{L}$ matrix. There is an inherent loss in the signal quality with each extra calculation hence reducing signal quality for $\boldsymbol{L}^{-1}$. The minimum SNR for $\boldsymbol{L}$ matrix is approximately 65dB, and approximately 30dB for $\boldsymbol{L}^{-1}$matrix. However, if we restrict condition number to less than 10^4 we can get an SNR better than 60dB for $\boldsymbol{L}^{-1}$matrix, hence satisfying the requirement of SNR greater than 50dB.

5.1.3. 4x4 Input Matrix

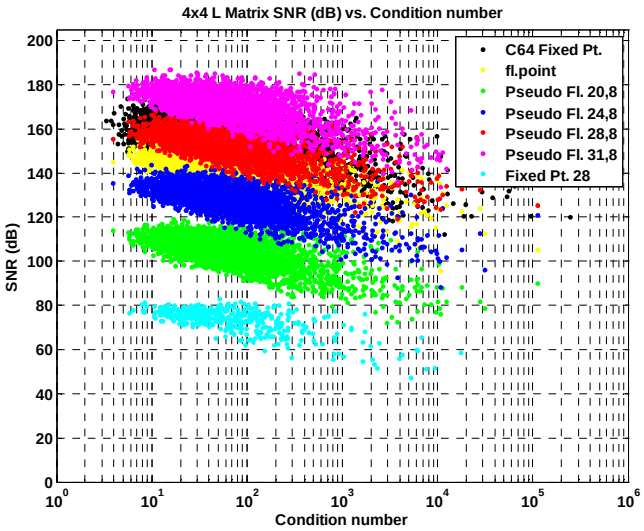

Figure 5.3: SNR(dB) for 4x4 L matrix

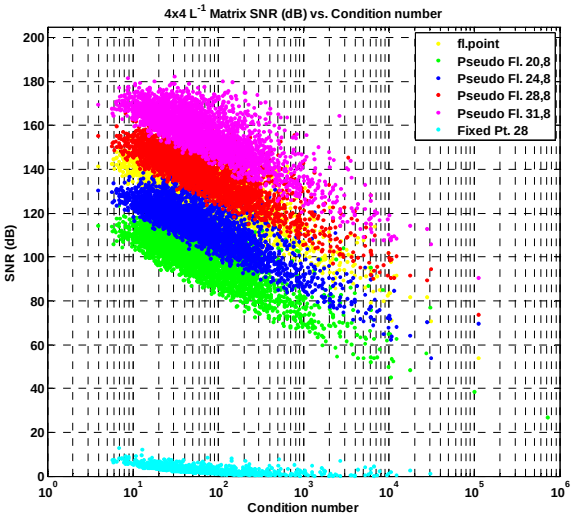

Figure 5.4: SNR(dB) for 4x4 L^{-1} matrix

Figure 5.3 shows the SNR in dB for 4×4 $\boldsymbol{L}$ matrix as a function of condition number (k) for C64x 32-bit fixed point, 32-bit floating point, 39-bit, 36-bit, 32-bit, and 28-bit pseudo floating point implementations. Whereas, Figure 5.4 shows the SNR in dB

for the corresponding $\boldsymbol{L}^{-1}$matrices. Following important observations are made from these two plots. Unlike the 2×2 case, the minimum condition number for the input coefficient matrix $\boldsymbol{A}$ is close to 4. This is a good condition number, as the dimensions of $\boldsymbol{A}$ are still relatively small. The worst condition number for $\boldsymbol{A}$ is observed to be around $2x10^5$, but the bulk of the condition numbers are in the range 4 to 10^3. It is noted that this range is bigger than the range of condition numbers for 2×2 matrices, because 4×4 matrices are slightly more complex than their 2×2 counterparts. The 32-bit fixed point provides higher SNR than the floating point implementation for the most part. However, there are cases where floating point performs equally as good. Unlike the 2×2 matrices, the 32-bit fixed point SNR is not close to 39-bit pseudo floating point, but closer to 36-bit pseudo floating point. However, 32-bit fixed point is still a good choice over 36-bit pseudo floating point. We see that SNR for 28-bit fixed point is below all other SNR plots. This is because fixed point representation loses SNR very rapidly, and runs into saturation problems very quickly. In this case, Cholesky decomposition fails at 27-bit fixed point precision. 27-bit fixed is unable to handle the range of numbers required for an MMSE MIMO receiver. Furthermore, 39-bit and 36-bit pseudo floating point implementations also give higher SNR than 32-bit floating point. SNRs for 32-bit and 28-bit pseudo floating point implementations are below 32-bit floating point. As before, the general slope of the SNRs for $\boldsymbol{L}^{-1}$ matrices is greater than that of $\boldsymbol{L}$ matrices. The minimum SNR for $\boldsymbol{L}$ matrix is approximately 70dB, and approximately 30dB for $\boldsymbol{L}^{-1}$matrix if we restrict condition number to less than 10^4. In this case if we want SNR for $\boldsymbol{L}^{-1}$to be greater than 50dB, then either we have to restrict condition numbers to less than 10^3, or pick 32-bit pseudo floating point implementation.

5.1.4. 8x8 Input Matrix

We make the following observations from SNR clouds in Figure 5.5 and Figure 5.6. The minimum condition number for the input coefficient matrix $\boldsymbol{A}$ is now around 50, which is much greater than corresponding numbers for the previous cases. The deterioration in the best possible condition number is now becoming non-trivial. The worst condition number for $\boldsymbol{A}$ is observed to be around 10^6, but most of the condition numbers are in the range 50 to 10^4.

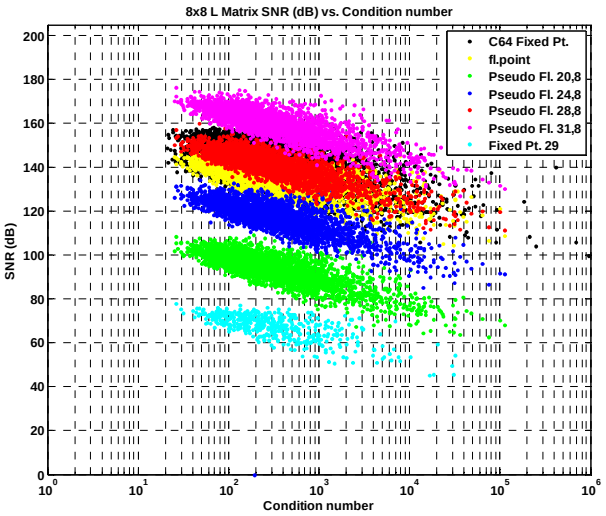

Figure 5.5: SNR(dB) for 8x8 L matrix

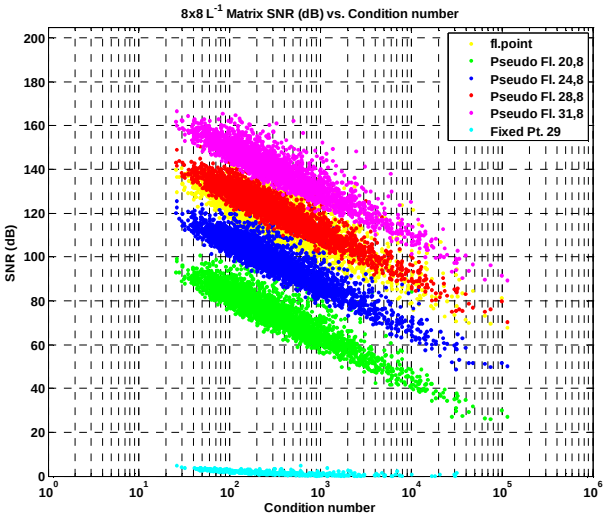

Figure 5.6: SNR(dB) for 8x8 L^{-1} matrix

Here we notice that the condition numbers are closely distributed within this range. The diversity in SNR for a given condition number is reduced due to increased complexity of 8×8 matrices. The 32-bit fixed point implementation converges towards 32-bit floating point, and 36-bit pseudo floating point implementations. Furthermore, SNR clouds for each pseudo floating point implementation show reduced scatter, and overlap minimally. SNRs for 32-bit and 28-bit pseudo floating point implementations are distinctly below 32-bit floating point. As before, the general slope of the SNRs for $\boldsymbol{L}^{-1}$ matrices is greater

than that of $\boldsymbol{L}$ matrices. This is consistent with the trend observed so far. The minimum SNR for $\boldsymbol{L}$ matrix is approximately 60dB, and approximately 20dB for $\boldsymbol{L}^{-1}$matrix. If a system specification relies on SNR for $\boldsymbol{L}^{-1}$matrix to be greater than 50dB, the choices are very limited. Either we restrict condition numbers to less than $2x10^3$ and choose 28-bit pseudo floating point implementation, or pick 32-bit pseudo floating point implementation.

5.1.5. 16x16 Input Matrix

16×16 is the maximum possible dimensions for our input matrix $\boldsymbol{A}$. Most MIMO systems do not use 16×16 matrices. We expect the SNR for these dimensions to be much worse than previous cases. Figure 5.7 and Figure 5.8 show the SNR clouds for 16×16 $\boldsymbol{L}$ and $\boldsymbol{L}^{-1}$ matrices. The minimum condition number for the input coefficient matrix $\boldsymbol{A}$ is greater than 100. The degradation in the best possible condition number is now evident. The worst condition number for $\boldsymbol{A}$ is observed to be around 10^7, and the condition numbers are almost uniformly distributed within the range 100 to $2x10^4$.

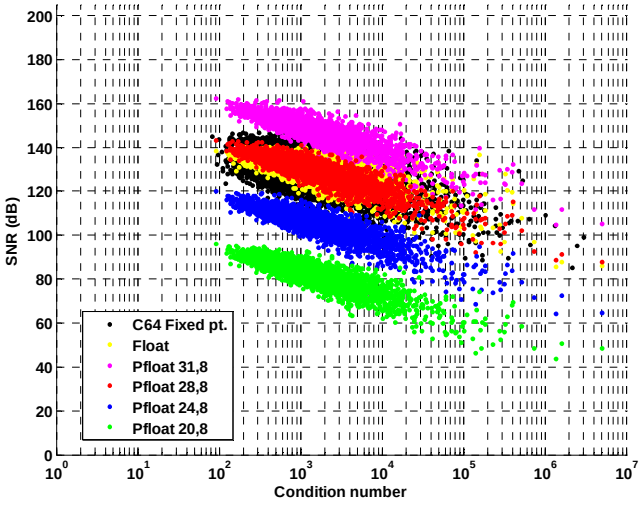

Figure 5.7: SNR(dB) for 16x16 L matrix

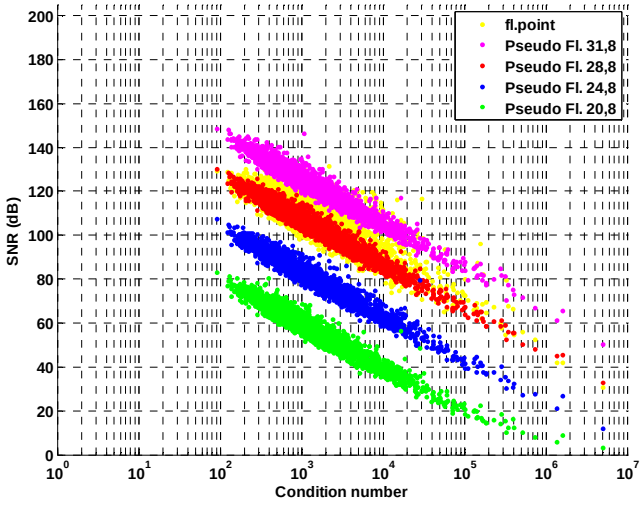

Figure 5.8: SNR(dB) for 16x16 L^{-1} matrix

The spread in SNR for a given condition number is narrowed down along the slope of the SNR cloud. This is attributed to the high dimensions and complexity of 16×16 matrices. The 32-bit fixed point implementation completely envelopes 32-bit floating point, and 36-bit pseudo floating point implementations. Furthermore, SNR clouds for each pseudo floating point implementation are non-overlapping, distinct and collimated. As expected the minimum SNR for $\boldsymbol{L}$ matrix is much lower than previous cases and falls at approximately 40dB. The SNR for $\boldsymbol{L}^{-1}$matrix is almost reduced to zero for the 28-bit pseudo floating point implementation. Here, we must choose 32-bit pseudo floating point implementation, and restrict condition numbers to less than $2x10^4$, if our system specification relies on SNR for $\boldsymbol{L}^{-1}$matrix to be greater than 50dB.

5.2. SNR for Solution Vector

In this section we discuss the SNR_{MMSE} results obtained for the solution vector. These results depend on modulating constellations. Also, we investigate the degradation in SNR'. Values of SNR_{line} for each constellation are selected according to Table 3.1. We expect these SNR_{line} values to be the usual lower and upper bounds for industry standards. We predict that these variables only depend on the condition number of the matrix $\boldsymbol{A}$. Since, SNR_{line} only controls the range of condition numbers, therefore, the results should not be affected by different values of SNR_{line}.

5.2.1. Simulation Specifications

For all simulation results shown in Section Chapter 0 the type of modulating constellation is very important, and it governs the loss in SNR' to some extent. Following variables are constant across all simulation results:

- Number of trials is set to 300
- M = 10
- Number of transmission antennas (N_T) = 4
- Number of receive antennas (N_R) = 4
- $Exponent = 8\ bits$

Variables varied across simulation results shown in Section Chapter 0 are given below:

- SNR_{line} is varied according to Table 3.1 for each constellation
- Signal variance (σ_S^2) is assigned the values 2^{22}, and 2^{24} for each SNR_{line}
- $Integer$ is given the values: 20, 16, 12, and 10

Since, both the number of transmission antennas (N_T), and the number of receive antennas (N_R) are fixed at 4, therefore, the input coefficient matrix $\boldsymbol{A}$ is a 4×4 matrix for all simulation results shown below. All SNR' values are calculated at $P_{sym} = 10^{-1}$.

5.2.2. QPSK SNR Analysis

Figure 5.9, and Figure 5.10, show SNR_{MMSE} (equation 3.1.1) for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $12dB$. Similarly, Figure 5.11, and Figure 5.12 show SNR_{MMSE} for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $50dB$. These plots show that there is negligible difference between values of SNR_{MMSE} as signal variance (σ_S^2) is varied from 2^{22} to 2^{24}. However, we observe that plots shown in Figure 5.9, and Figure 5.10, are subsets of plots given in

Figure 5.11, and Figure 5.12 respectively. This shows that SNR_{MMSE} is independent of SNR_{line}.

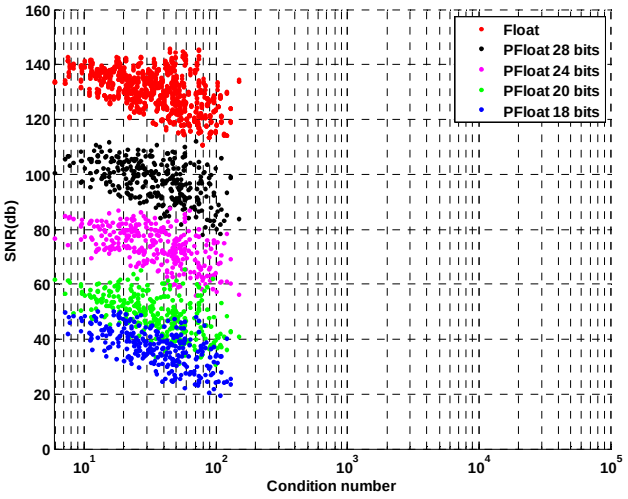

Figure 5.9: QPSK Solution Vector *SNR* for SNR_{line} = 12dB and Signal Variance = 2^{22}

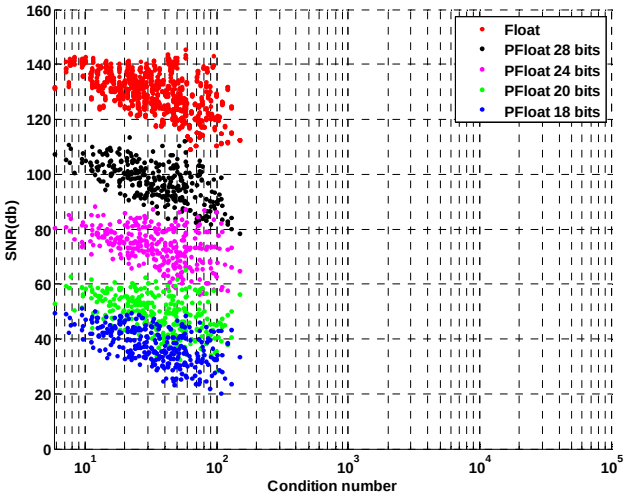

Figure 5.10: QPSK Solution Vector *SNR* for SNR_{line} = 12dB and Signal Variance = 2^{24}

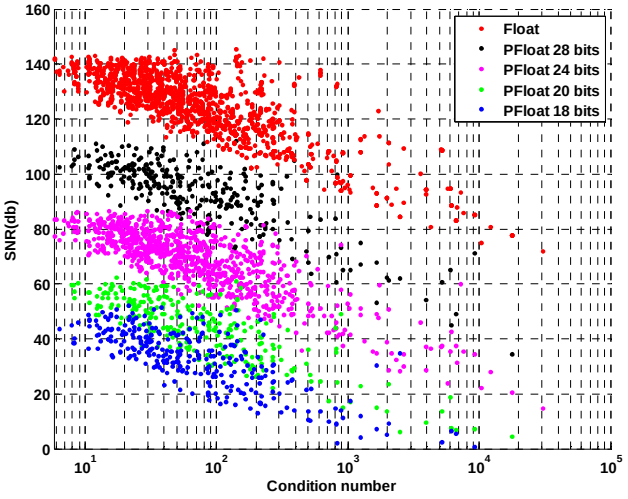

Figure 5.11: QPSK Solution Vector *SNR* for SNR_{line} = 50dB and Signal Variance = 2^{22}

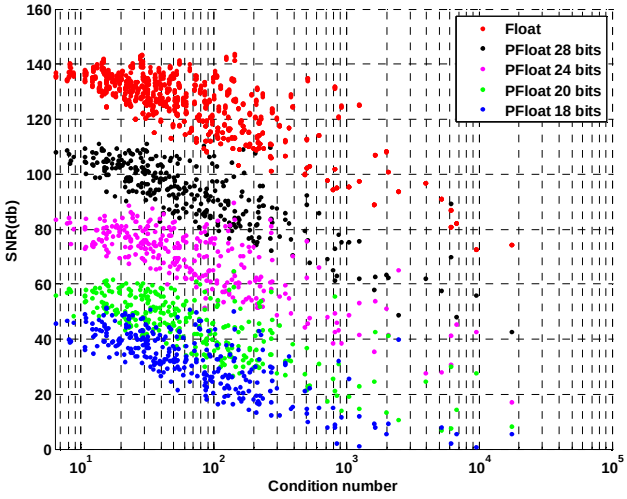

Figure 5.12: QPSK Solution Vector *SNR* for SNR_{line}= 50dB and Signal Variance = 2^{24}

These results confirm the initial predictions that the behavior of the linear equation (1.2.1) depends on condition number only, and is independent of the range of condition numbers defined by SNR_{line}. As expected, SNR_{MMSE} degrades with a decrease in precision bits.

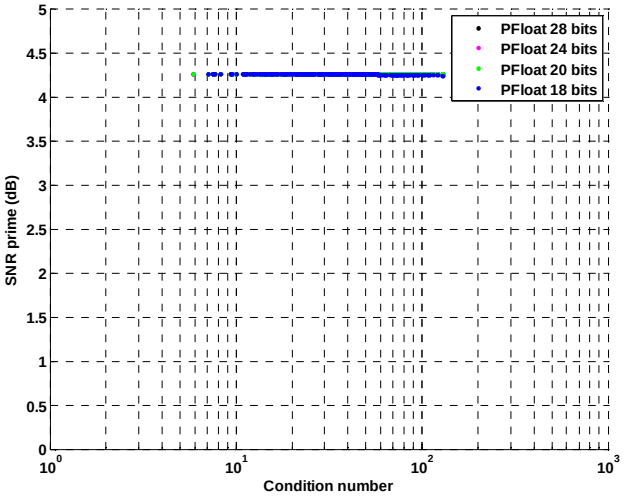

Figure 5.13: QPSK SNR'(dB) for SNR_{line}= 12dB and Signal Variance = 2^{22}

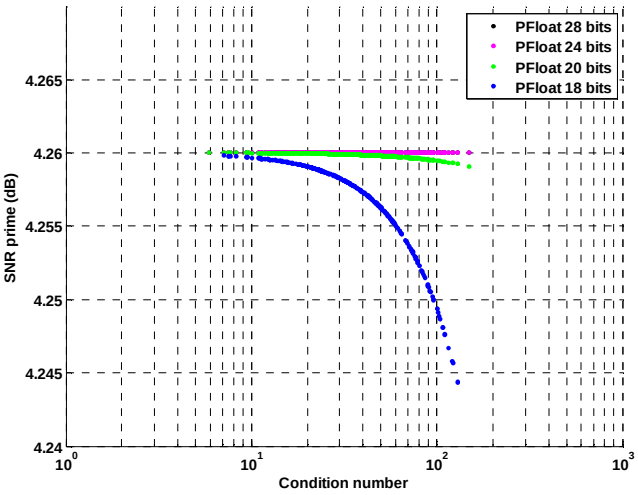

Figure 5.14: QPSK SNR'(dB) for SNR_{line}= 12dB and Signal Variance = 2^{22} (zoom)

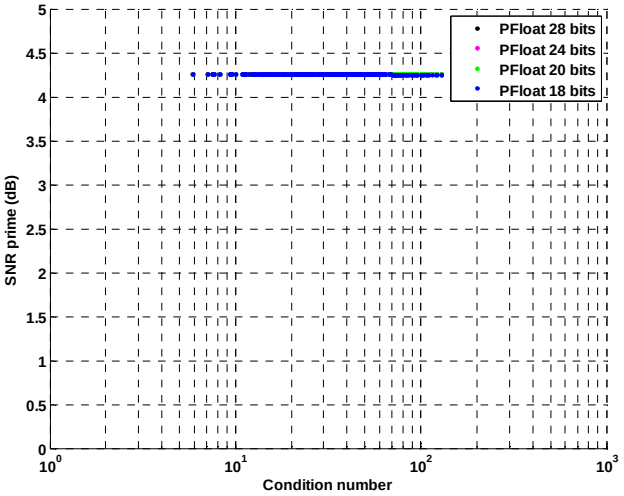

Figure 5.15: QPSK SNR'(dB) for SNR_{line}= 12dB and Signal Variance = 2^{24}

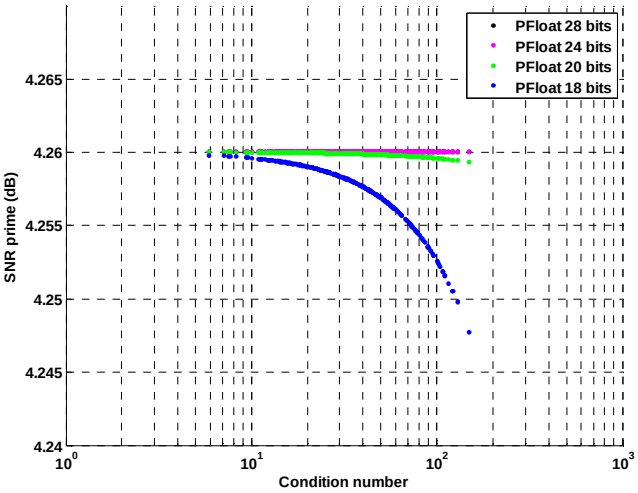

Figure 5.16: QPSK SNR'(dB) for SNR_{line}= 12dB and Signal Variance = 2^{24} (zoom)

Figure 5.13 and Figure 5.15 show plots for SNR' (equation 3.1.4) for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $12dB$. We notice that the base value for SNR' is $4.26dB$. This is consistent with the standard SER plots shown in Figure 4.1. Figure 4.1 shows that SNR corresponding to $P_{sym} = 10^{-1}$ should be around $4dB$. Our calculations reveal that the base value for SNR' is indeed close to $4dB$, as expected. In Figure 5.13 and Figure 5.15, the variations in SNR', for various bit

precisions, along the y-axis cannot be distinguished. Therefore, Figure 5.14 and Figure 5.16 are presented with data zoomed along the y-axis. At this zoom level, significant variations across bit precisions are visible. Even at this scale, any variation in values of SNR' for 28-bit and 24-bit precisions is still un-distinguishable. However, 20-bit precision shows a very small deviation (~0.005) from base value (4.26), and 18-bit precision shows slightly higher deviation (~0.015) in SNR' with increasing condition numbers. It is further observed that there is a greater loss in SNR' for $\sigma_S^2 = 2^{22}$ than $\sigma_S^2 = 2^{24}$ for 18-bit precision. It is interesting, as this is the first time that we have been able to distinguish a different behavior for the two signal variances used. From section 3.1 we know that $\boldsymbol{A} = \boldsymbol{H}_e^{\dagger}\boldsymbol{H}_e + \sigma_N^2\boldsymbol{I}_{N_T}$, and noise variance is defined as $\sigma_N^2 = \sigma_S^2/SNR_{line}$. This shows that σ_S^2 controls the noise added to the diagonal of $\boldsymbol{A}$. A bigger signal variance will add a bigger number to the diagonal $\boldsymbol{A}$, hence improving the condition of $\boldsymbol{A}$, and in turn improving the overall signal power. From these observations we infer that a higher signal variance gives better SNR than a lower signal variance.

Figure 5.17 and Figure 5.18 show plots for SNR' (equation 3.1.4) for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $50dB$. Significant variations in SNR', for various bit precisions are easily distinguished. Any variation in values of SNR' for 28-bit and 24-bit precisions is still un-distinguishable. However, 20-bit and 18-bit precisions show noticeable degradation in SNR' with increasing condition numbers. SNR' for 20-bit pseudo floating point starts to degrade close to condition number 10^3, but SNR' for 18-bit pseudo floating point starts to degrade around condition number 10^2.

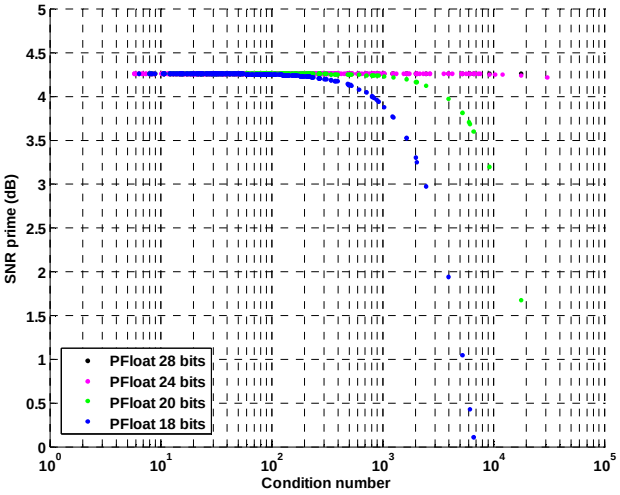

Figure 5.17: QPSK SNR'(dB) for SNR_{line}= 50dB and Signal Variance = 2^{22}

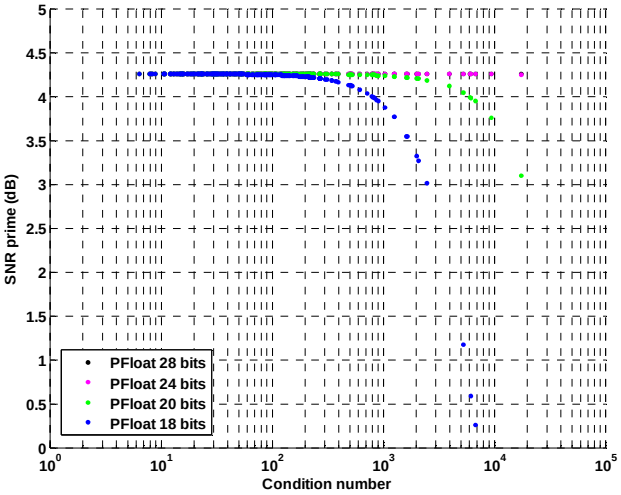

Figure 5.18: QPSK SNR'(dB) for SNR_{line}= 50dB and Signal Variance = 2^{24}

Therefore a system that operates at $SNR_{line} = 50dB$ will be significantly affected by a lower precision system design. In other words 20-bit precision can be considered a cut-off point for systems with $SNR_{line} = 50dB$. From all the figures shown in section 5.2.2, it is observed that SNR' behaves independently of SNR_{line}. This is evident from the fact that Figure 5.13 and Figure 5.15 appear to be subsets of figures Figure 5.17 and Figure 5.18. This supports the initial expectations, as SNR_{line} only controls the range of condition numbers, and has no effect on the value of SNR'.

5.2.3. 16QAM SNR Analysis

Next we discuss SNR_{MMSE} (equation 3.1.1) for signal variances (σ_S^2) 2^{22} and 2^{24} when SNR_{line} is fixed at $18dB$ for 16QAM constellation. The corresponding plots are shown in Figure 5.19, and, Figure 5.20 respectively. Similarly, Figure 5.21, and Figure 5.22 show SNR_{MMSE} for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $50dB$. As in the QPSK case these plots show negligible difference between SNR_{MMSE} values as signal variance (σ_S^2) is varied from 2^{22} to 2^{24}.

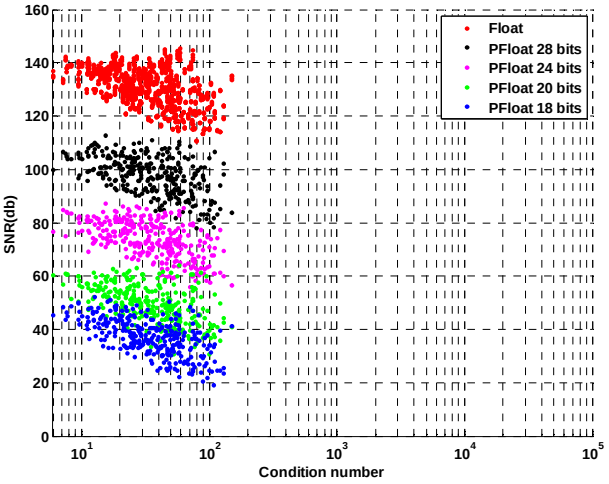

Figure 5.19: 16QAM Solution Vector SNR for SNR_{line}= 18dB and Signal Variance = 2^{22}

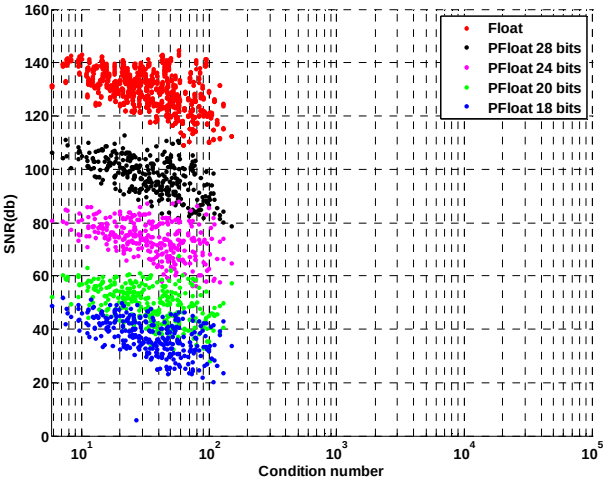

Figure 5.20: 16QAM Solution Vector SNR for SNR_{line}= 18dB and Signal Variance = 2^{24}

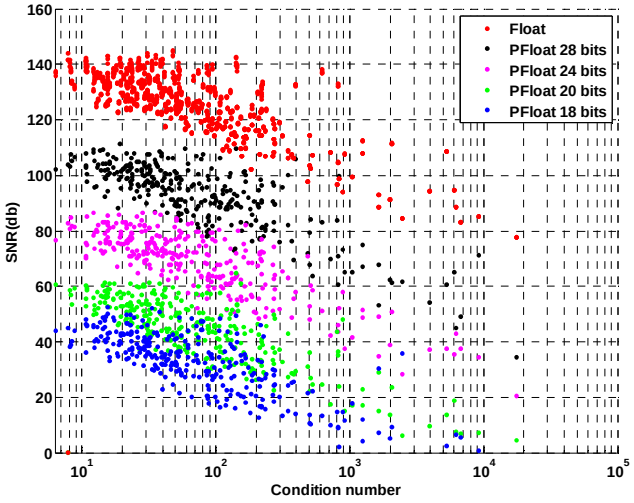

Figure 5.21: 16QAM Solution Vector SNR for SNR_{line}= 50dB and Signal Variance = 2^{22}

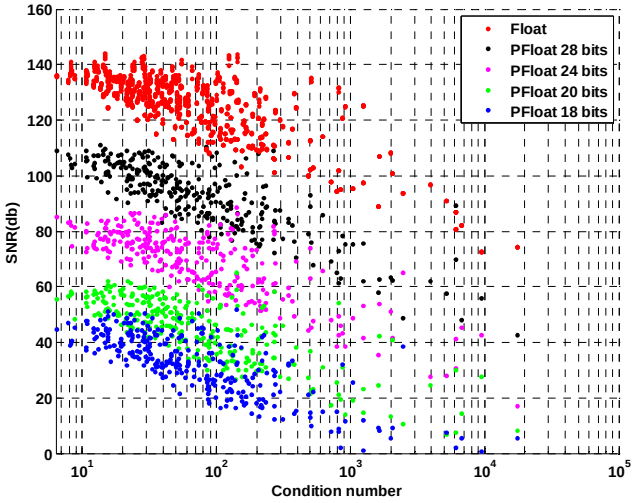

Figure 5.22: 16QAM Solution Vector SNR for SNR_{line}= 50dB and Signal Variance = 2^{24}

Similar to QPSK results, we observe that plots shown in Figure 5.19, and Figure 5.20, are simply smaller subsections of plots given in Figure 5.21, and Figure 5.22 respectively. Again these results show that SNR_{MMSE} is independent of SNR_{line}, and confirm the initial predictions that the behavior of the linear equation (1.2.1) depends on condition number only, and is independent of the range of condition numbers defined by SNR_{line}.

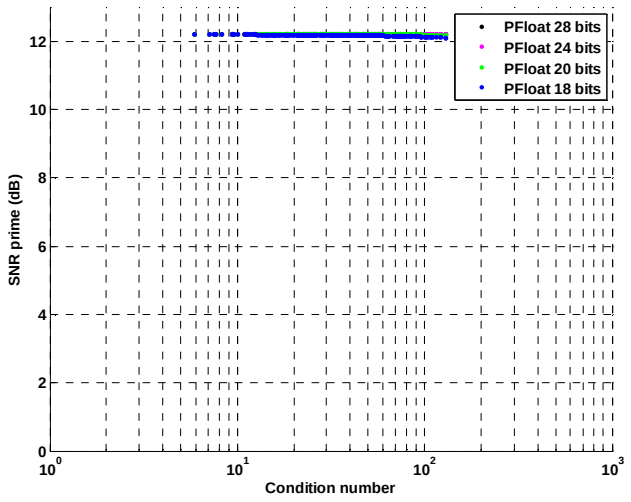

Figure 5.23: 16QAM SNR'(dB) for SNR_{line}= 18dB and Signal Variance = 2^{22}

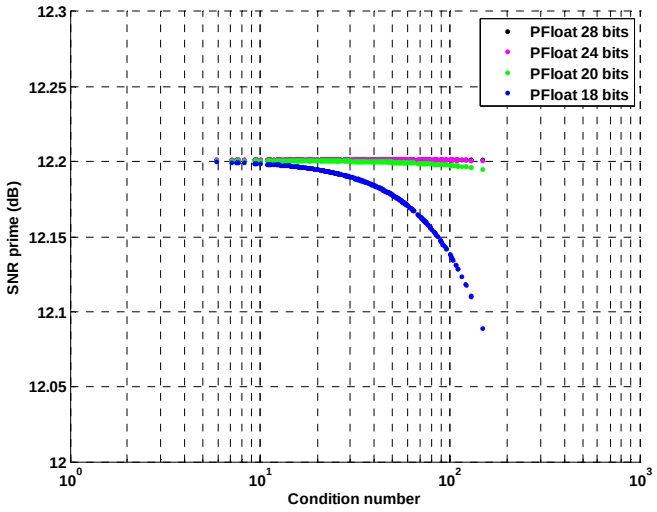

Figure 5.24: 16QAM SNR'(dB) for SNR_{line}= 18dB and Signal Variance = 2^{22} (zoom)

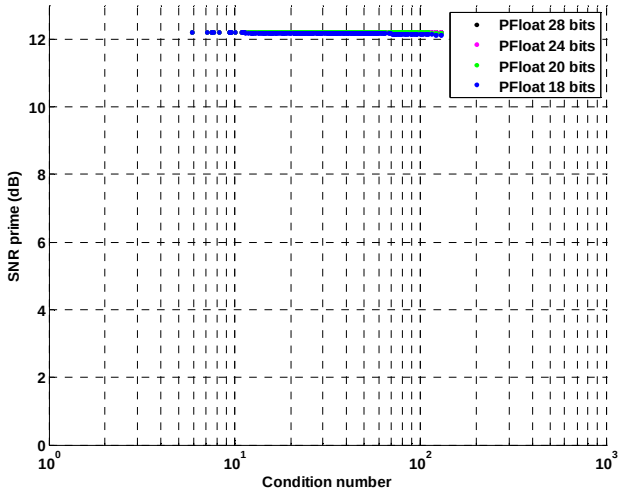

Figure 5.25: 16QAM SNR'(dB) for SNR_{line}= 18dB and Signal Variance = 2^{24}

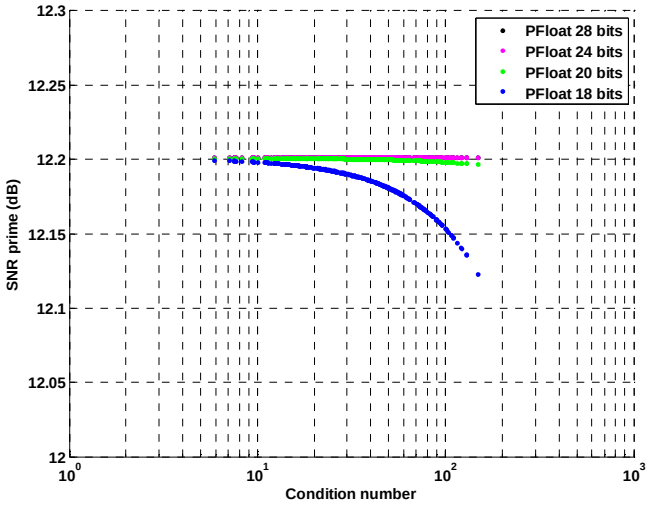

Figure 5.26: 16QAM SNR'(dB) for SNR_{line}= 18dB and Signal Variance = 2^{24} (zoom)

Figure 5.23 and Figure 5.25 show plots for SNR' (equation 3.1.4) for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $18dB$. We notice that the base value for SNR' is $12.2dB$ as opposed to $4.26dB$ for the QPSK case. This is consistent with data given in Figure 4.2. In Figure 4.2 we see that SNR corresponding to $P_{sym} = 10^{-1}$is approximately $12dB$. So, we expect the base value for SNR to be close to

$12dB$. At the scale used to plot Figure 5.23 and Figure 5.25, the variations in SNR', for various bit precisions, along the y-axis are barely distinguishable. However, Figure 5.24 and Figure 5.26 are presented so the deviations in SNR' can be observed more closely. Even though plots in Figure 5.24 and Figure 5.26 show significant variations across bit precisions, it should be noted that the maximum variation is no larger than 0.08 for the lowest bit precision of 18-bits. Also, any variation in SNR' values for 28-bit and 24-bit precisions is still un-distinguishable, as we observed in the QPSK case. It can be safely concluded that these variations are within acceptable limits from a practical industrial point of view, as they are very close to the base value of $12.2dB$. Similar to the QPSK case, we observe that there is a greater loss in SNR' for $\sigma_S^2 = 2^{22}$ than $\sigma_S^2 = 2^{24}$ for 18-bit precision. Once again we are able to distinguish a different behavior for the two signal variances used. As described in the QPSK case, a signal variance of 2^{24} adds a bigger number to the diagonal $\boldsymbol{A}$, hence improving the condition of $\boldsymbol{A}$, and in turn improving the overall signal power.

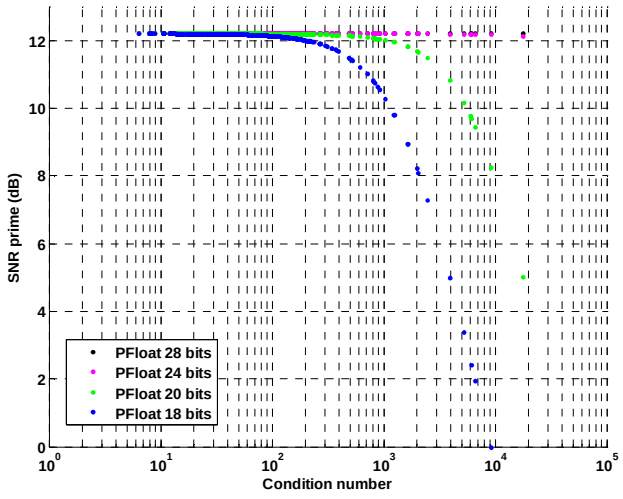

Figure 5.27: 16QAM SNR'(dB) for SNR_{line}= 50dB and Signal Variance = 2^{22}

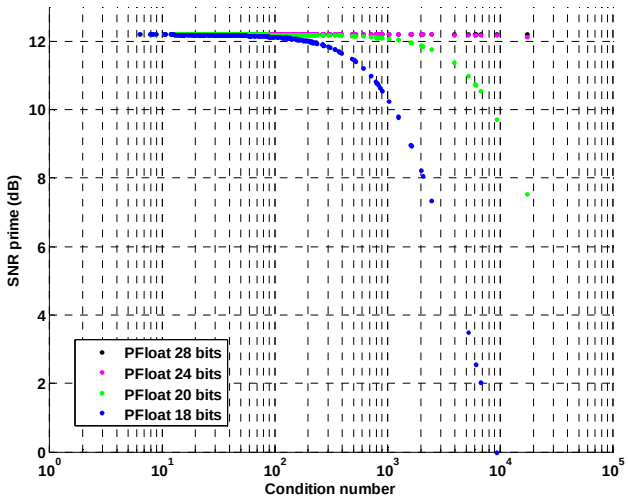

Figure 5.28: 16QAM SNR'(dB) for SNR_{line}= 50dB and Signal Variance = 2^{24}

Figure 5.27 and Figure 5.28 show plots for SNR' (equation 3.1.4) for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $50dB$. Significant variations in SNR', for various bit precisions are easily distinguished. Variation in SNR' values for 28-bit and 24-bit precisions is still un-distinguishable. However, 20-bit and 18-bit precisions show noticeable degradation in SNR' with increasing condition numbers. SNR' for 20-bit pseudo floating point starts to degrade close to condition number $5x10^2$, but SNR' for 18-bit pseudo floating point starts to degrade around condition number $7x10^1$. Notice that this degradation starts at an earlier condition number than the QPSK case. This is because of the inherent behavior of 16QAM constellation points. The constellation points are packed closer together than QPSK, and are hence more susceptible to noise. Again we must conclude that 20-bit precision can be considered a cut-off point for systems with $SNR_{line} = 50dB$, as the signal degradation is significant as condition numbers go above 10^2. From all the figures shown in section 5.2.3, it is observed that SNR' behaves independently of SNR_{line}. This is evident from the fact that Figure 5.23 and Figure 5.25 appear to be subsets of Figure 5.27 and Figure 5.28. This supports the initial expectations, as SNR_{line} only controls the range of condition numbers, and has no effect on the value of SNR'.

5.2.4. 64QAM SNR Analysis

SNR_{MMSE} (equation 3.1.1) plots for signal variances (σ_S^2) 2^{22} and 2^{24} for 64QAM constellation when SNR_{line} is fixed at $24dB$ are shown in Figure 5.29, and, Figure 5.30 respectively. Also, Figure 5.31, and Figure 5.32 show SNR_{MMSE} for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $50dB$. As in the QPSK, and 16QAM cases these plots show negligible difference between SNR_{MMSE} values as signal variance

(σ_S^2) is varied from 2^{22} to 2^{24}. Also, consistent with QPSK, and 16QAM results, we observe that plots shown in Figure 5.29, and, Figure 5.30, are simply subsets of plots given in Figure 5.31, and Figure 5.32 respectively.

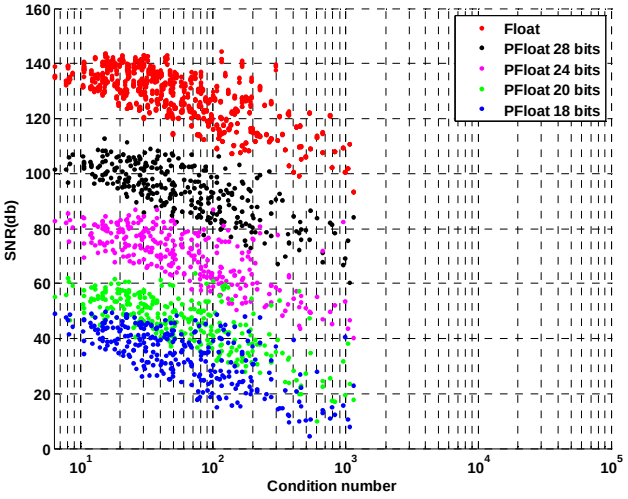

Figure 5.29: 64QAM Solution Vector *SNR* for SNR_{line}= 24dB and Signal Variance = 2^{22}

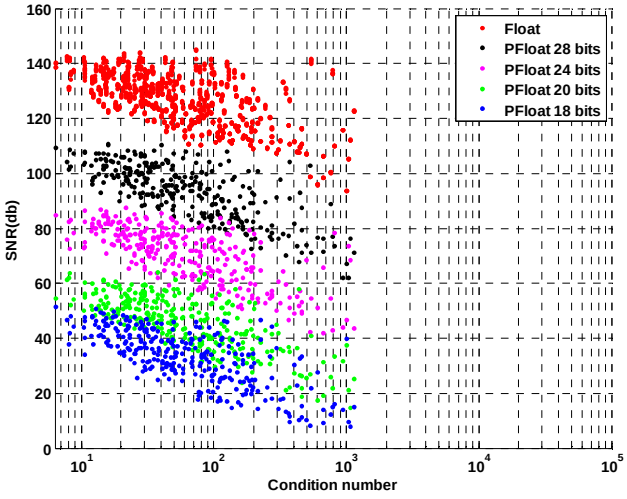

Figure 5.30: 64QAM Solution Vector *SNR* for SNR_{line}= 24dB and Signal Variance = 2^{24}

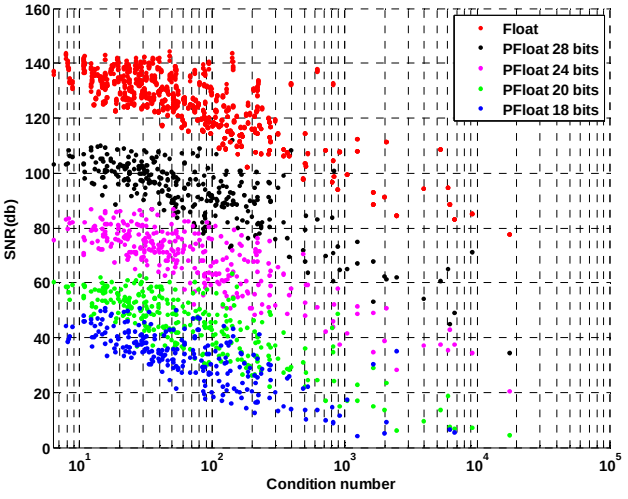

Figure 5.31: 64QAM Solution Vector *SNR* for SNR_{line}= 50dB and Signal Variance = 2^{22}

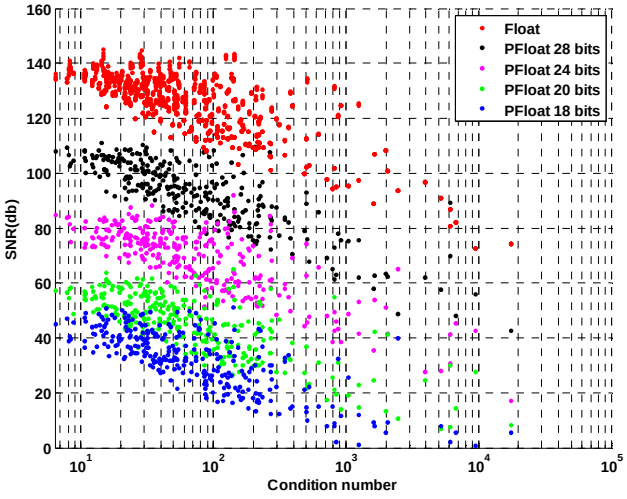

Figure 5.32: 64QAM Solution Vector *SNR* for SNR_{line}= 50dB and Signal Variance = 2^{24}

Again these results show that SNR_{MMSE} is independent of SNR_{line}, and confirm the initial predictions that the behavior of the linear equation (1.2.1) depends on condition number only, and is independent of the range of condition numbers defined by SNR_{line}.

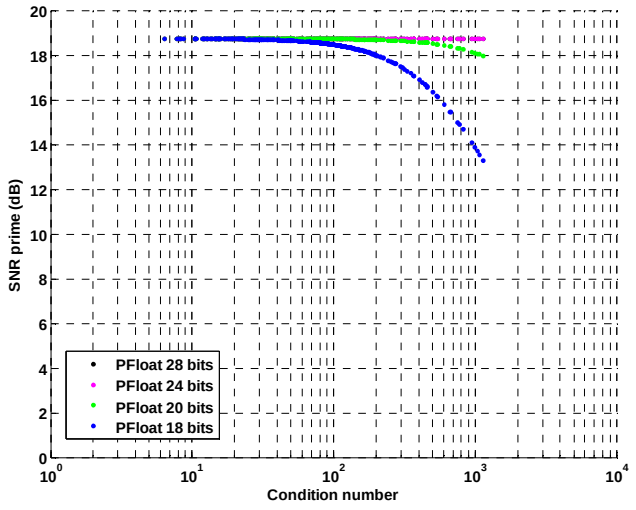

Figure 5.33: 64QAM SNR'(dB) for SNR_{line}= 24dB and Signal Variance = 2^{22}

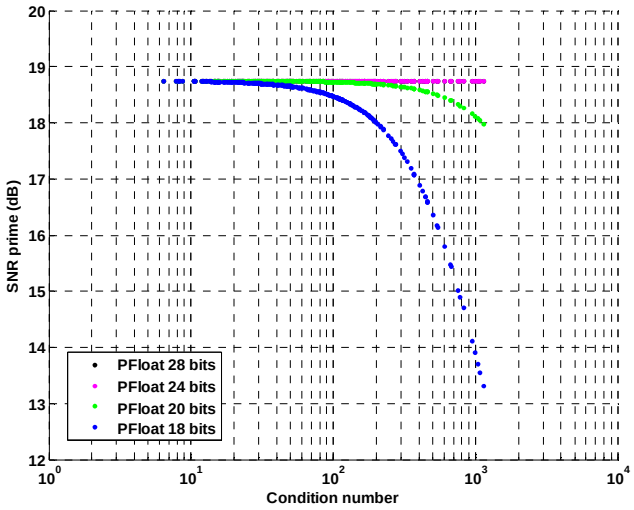

Figure 5.34: 64QAM SNR'(dB) for SNR_{line}= 24dB and Signal Variance = 2^{22} (zoom)

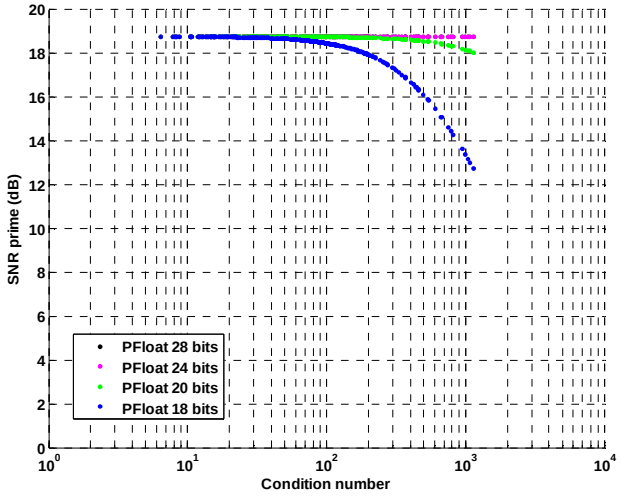

Figure 5.35: 64QAM SNR'(dB) for SNR_{line}= 24dB and Signal Variance = 2^{22}

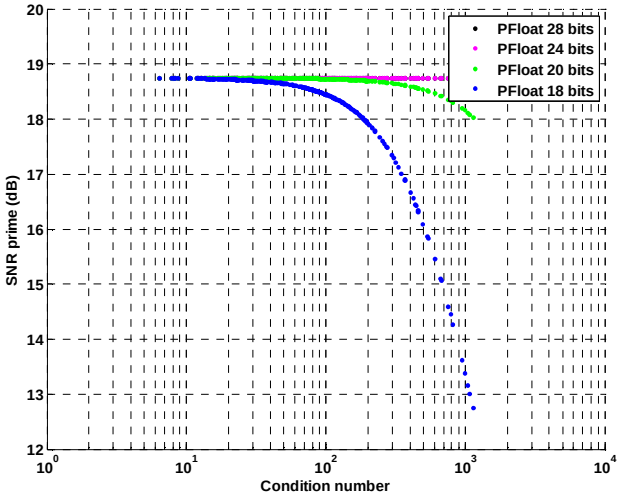

Figure 5.36: 64QAM SNR'(dB) for SNR_{line}= 24dB and Signal Variance = 2^{22} (zoom)

Next we show plots for SNR' (equation 3.1.4) for 64QAM constellation points for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $24dB$, in Figure 5.33 and Figure 5.35. We notice that the base value for SNR' is $18.755dB$, which is

higher than both $4.26dB$ and $12.2dB$ from QPSK and 16QAM cases respectively. This is consistent with data given in Figure 4.3. In Figure 4.3 we see that SNR corresponding to $P_{sym} = 10^{-1}$is a little less than $19dB$. So, we expect the base value for SNR to be close to $18.8dB$. At the scale used to plot Figure 5.33 and Figure 5.35, the variations in SNR', for various bit precisions, along the y-axis are considerably distinguishable as compared to the QPSK and 16QAM cases. To be consistent with the data provided for QPSK and 16QAM, Figure 5.34 and Figure 5.36 are presented so the deviations in SNR' can be observed more closely. Plots in Figure 5.24 and Figure 5.26 show significant variations across bit precisions, and it is noted that the maximum variation is around 6 for the lowest bit precision of 18-bits. This is a much greater variation than the QPSK and 16QAM cases. Even though there is significant degradation in SNR' values for 18-bit and 20-bit precisions, any variation for 28-bit and 24-bit precisions is still un-distinguishable. We can infer from these results that 64QAM loses signal quality much more rapidly at lower precisions than QPSK and 16QAM. In the case of 64QAM we cannot conclude that these variations are within acceptable limits from a practical industrial point of view, as it will greatly depend on the tolerance of the system whether it can handle a degradation of 6dB SNR or not. Similar to the QPSK and 16QAM cases, we observe that there is a greater loss in SNR' for $\sigma_S^2 = 2^{22}$ than $\sigma_S^2 = 2^{24}$ for 18-bit precision. The difference is SNR' values is approximately $0.5dB$. This may be significant to a system that is sensitive to lower values of SNR'.

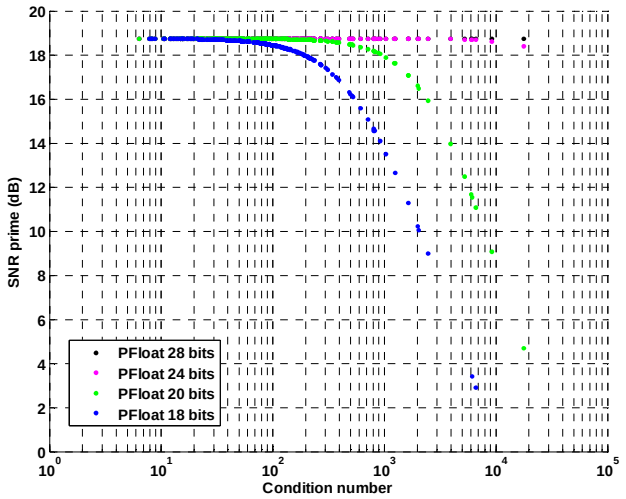

Figure 5.37: 64QAM SNR'(dB) for SNR_{line}= 50dB and Signal Variance = 2^{22}

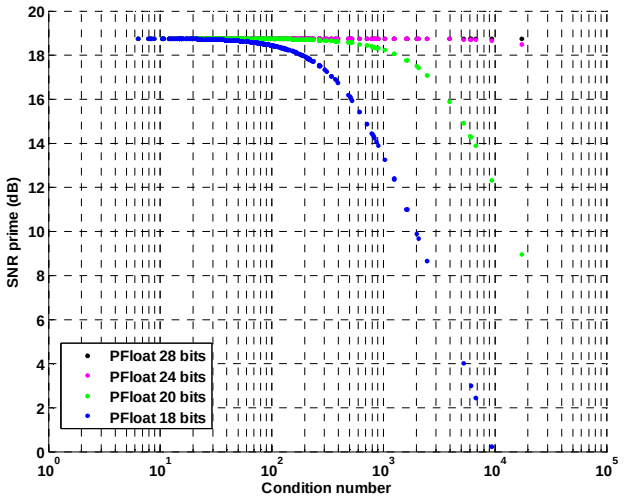

Figure 5.38: 64QAM SNR'(dB) for SNR_{line}= 50dB and Signal Variance = 2^{24}

Figure 5.37 and Figure 5.38 show plots for SNR' (equation 3.1.4) for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $50dB$. Significant variations in SNR', for various bit precisions are easily distinguished. Here, variation in SNR' values for 28-bit and 24-bit precisions are distinguishable. As in the case of QPSK and 16QAM 20-bit and 18-bit precisions show noticeable degradation in SNR' with increasing condition numbers. We notice that SNR' actually comes very close to zero for 18-bit precision in Figure 5.38. SNR' for 20-bit pseudo floating point starts to degrade close to condition number $2x10^2$, but SNR' for 18-bit pseudo floating point starts to degrade around condition number $3x10^1$. Notice that this degradation starts at an earlier condition number than both QPSK and 16QAM cases. This is because of the inherent behavior of 64QAM constellation points. The constellation points are packed closer together than 16QAM and QPSK, and are hence more susceptible to noise. Here conclude that 24-bit precision should be adopted for systems with $SNR_{line} = 50dB$, as the signal degradation is significant for bit precisions lower than 24-bits, as condition

numbers go above 10^2. From all the figures shown in section 5.2.4, it is observed that SNR' behaves independently of SNR_{line}. This is evident from the fact that Figure 5.33 and Figure 5.35 appear to be subsets of Figure 5.37 and Figure 5.38. This supports the initial expectations, as SNR_{line} only controls the range of condition numbers, and has no effect on the value of SNR'.

5.3. Loss in Symbol Error Rate

In this section we study loss in symbol error rate (SER) by analyzing the ratio P_{sym}/P'_{sym}. As discussed in section 4.1 P_{sym} is a function of SNR. With the new SNR' values calculated in the previous section, we now have the data to get the corresponding P'_{sym} values at each bit precision. As before, values of SNR_{line} for each constellation are selected according to Table 3.1. We expect these SNR_{line} values to be the usual lower and upper bounds for industry standards. As observed in section 5.2, SNR' values were independent of SNR_{line}. Based on this observation, we predict that the ratio P_{sym}/P'_{sym} depends on the condition number of the matrix $\boldsymbol{A}$ only, and is independent of SNR_{line}.

5.3.1. Simulation Specifications

For all simulation results shown in Section 5.3 the type of modulating constellation is very important, and it governs the loss in symbol error rate to some extent. Following variables are constant across all simulation results:

- Number of trials is set to 300
- M = 10
- Number of transmission antennas (N_T) = 4

- Number of receive antennas (N_R) = 4
- $Exponent = 8\ bits$

Variables varied across simulation results shown in Section Chapter 0 are given below:

- SNR_{line} is varied according to Table 3.1 for each constellation
- Signal variance (σ_S^2) is assigned the values 2^{22}, and 2^{24} for each SNR_{line}
- $Integer$ is given the values: 20, 16, 12, and 10

Since, both the number of transmission antennas (N_T), and the number of receive antennas (N_R) are fixed at 4, therefore, the input coefficient matrix $\boldsymbol{A}$ is a 4×4 matrix for all simulation results shown below. All calculations for the ratio P_{sym}/P'_{sym} are done with $P_{sym} = 10^{-1}$.

5.3.2. QPSK SER Loss

Figure 5.39 and Figure 5.41 show loss in SER for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $12dB$ for a QPSK system. These plots show a nearly perfect ratio of $P_{sym}/P'_{sym} = 1.0$ at all bit precisions. However, if we zoom in along the y-axis, we see that there is some loss in SER for 18-bits, and an almost negligible loss in SER for 20-bits (Figure 5.40 and Figure 5.42). No loss in SER is observed for 24-bit and 28-bit precisions in any of the figures shown above. Also, we notice that the maximum loss in SER is still very small for 18-bit

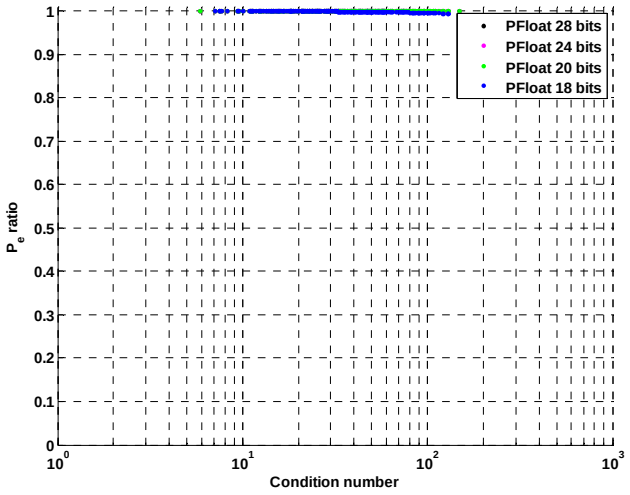

Figure 5.39: QPSK Pe/Pe' for SNR_{line}= 12dB and Signal Variance = 2^{22}

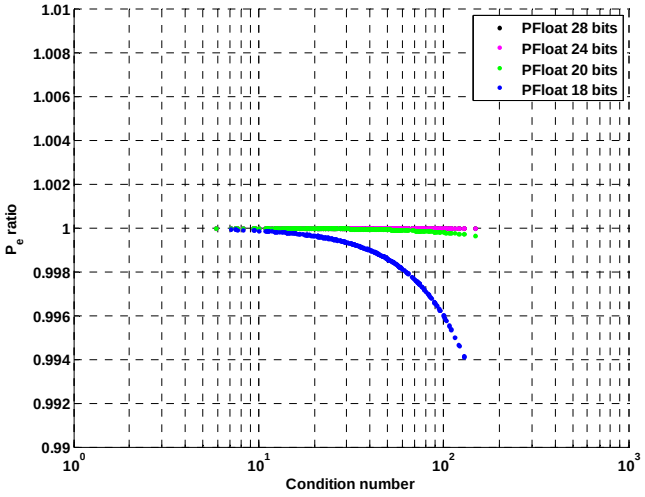

Figure 5.40: QPSK Pe/Pe' for SNR_{line}= 12dB and Signal Variance = 2^{22} (zoom)

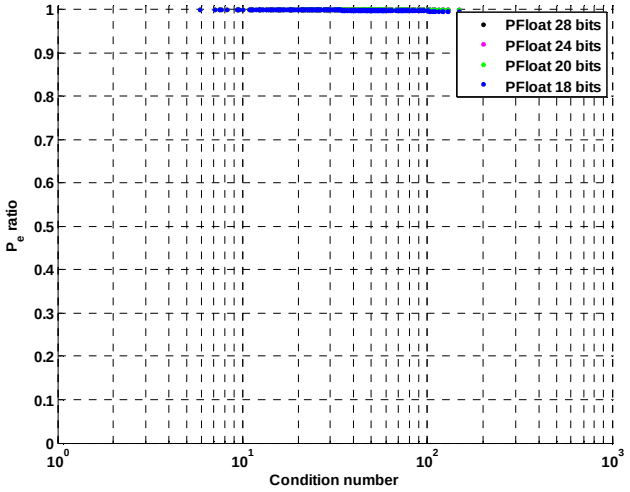

Figure 5.41: QPSK Pe/Pe' for SNR_{line}= 12dB and Signal Variance = 2^{24}

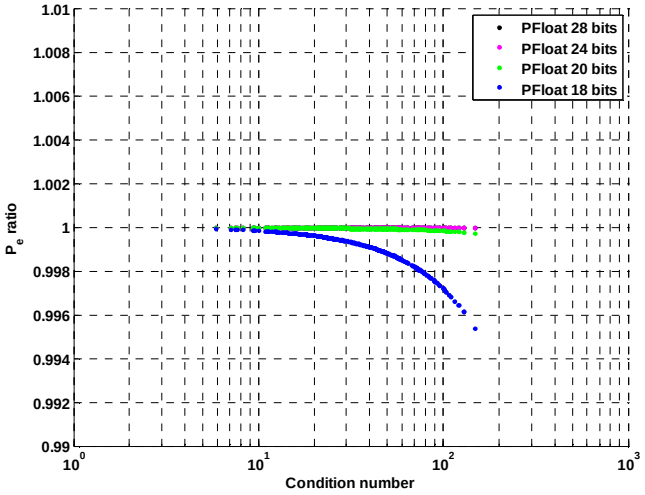

Figure 5.42: QPSK Pe/Pe' for SNR_{line}= 12dB and Signal Variance = 2^{24} (zoom)

precision (~0.006) with increasing condition numbers. It is further observed that there is a greater loss in SER for $\sigma_S^2 = 2^{22}$ than $\sigma_S^2 = 2^{24}$ for 18-bit precision. This is in accordance with previous results of SNR' in section 5.2. As discussed in section 4.1, we know that $P'_{sym} = f(SNR')$, and hence $P'_{sym} \propto SNR'$. If a system is to be designed with condition numbers less than $2x10^2$ then any bit precision will be acceptable.

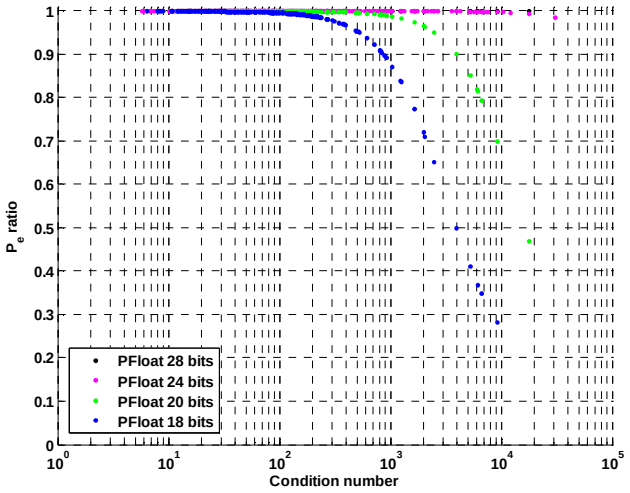

Figure 5.43: QPSK Pe/Pe' for SNR_{line}= 50dB and Signal Variance = 2^{22}

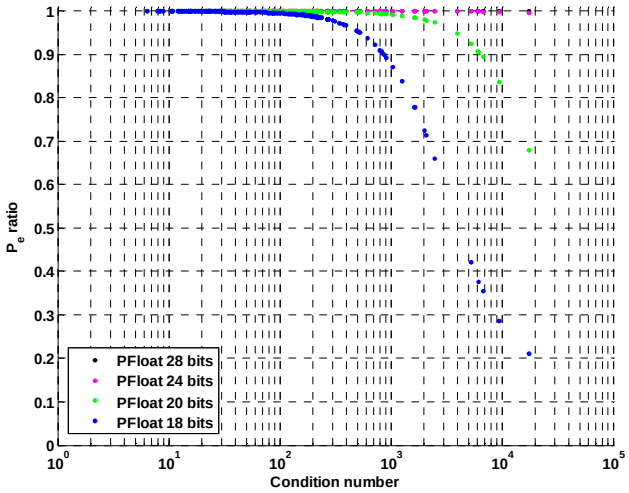

Figure 5.44: QPSK Pe/Pe' for SNR_{line}= 50dB and Signal Variance = 2^{24}

Figures 5.43 and 5.44 show loss in SER for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $50dB$ for a QPSK system. These plots show a clear degradation of the ratio P_{sym}/P'_{sym} at all bit precisions except for 28-bits. Even at the given scale it is clear that there is a significant loss in SER for 18-bits. Loss in SER for 20-bits starts at a later condition number, but similar to the 18-bits case, the ratio degrades very rapidly. Minor loss in SER is observed for 24-bits. No loss in SER is observed for 28-bit precisions in any of the figures shown above. These results are as expected. The SER is expected to degrade with increase in condition number. The degradation should be more drastic for lower bit precisions than higher ones. As observed the SER stay stable and close to perfect with 28-bit precision even when the condition number goes above $10^{4.}$

5.3.3. 16QAM SER Loss

Figure 5.45 and Figure 5.47 show loss in SER for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $18dB$ for a 16QAM system. Unlike the QPSK

case, these plots do not show a nearly perfect ratio of $P_{sym}/P'_{sym} = 1.0$ at all bit precisions. There is a noticeable difference in the P_{sym}/P'_{sym} ratio across the tested bit precisions at higher condition numbers. If we zoom in along the y-axis, we see that there is some loss in SER for 18-bits, and an almost negligible loss in SER for 20-bits (Figure 5.46 and Figure 5.48). No loss in SER is observed for 24-bit and 28-bit precisions in any of the figures shown below. Also, we notice that the maximum loss in SER is still very small for 18-bit precision (~0.005) with increasing condition numbers. It is further observed that there is a greater loss in SER for $\sigma_S^2 = 2^{22}$ than $\sigma_S^2 = 2^{24}$ for 18-bit precision. This is in accordance with previous results of SNR' in section 5.2. As discussed in section 4.1, we know that $P'_{sym} = f(SNR')$, and hence $P'_{sym} \propto SNR'$. If a system is to be designed with condition numbers less than 2x10^2 then any bit precision will be acceptable.

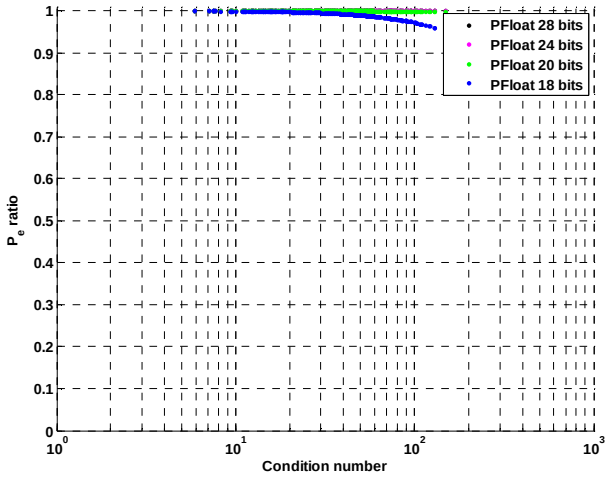

Figure 5.45: 16QAM Pe/Pe' for SNR_{line}= 18dB and Signal Variance = 2^{22}

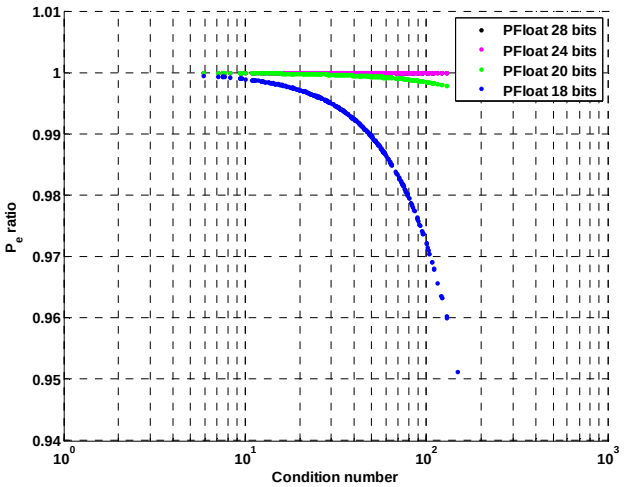

Figure 5.46: 16QAM Pe/Pe' for SNR_{line}= 18dB and Signal Variance = 2^{22} (zoom)

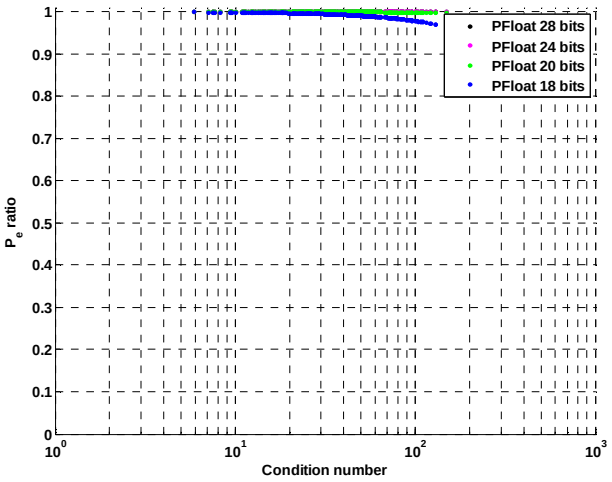

Figure 5.47: 16QAM Pe/Pe' for SNR_{line}= 18dB and Signal Variance = 2^{24}

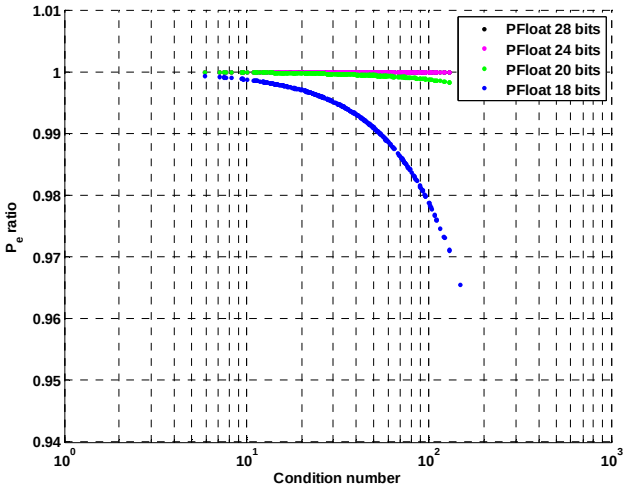

Figure 5.48: 16QAM Pe/Pe' for SNR_{line}= 18dB and Signal Variance = 2^{24} (zoom)

Figures 5.49 and 5.50 show loss in SER for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $50dB$ for a 16QAM system. These plots show a clear degradation of the ratio P_{sym}/P'_{sym} at all bit precisions. The degradation for 28-bits is very small, but it is noticeable for signal variance of 2^{22}. It is clear that there is a significant loss in SER for both 18-bits and 20-bits. Loss in SER for 24-bits starts at a later condition number, and surprisingly does not degrade very quickly. Minor loss in SER is observed for 28-bits at very high condition numbers. These results show that 24-bits is the cut-off point for a 16QAM system, if we want to get reasonable reliability even at higher condition numbers. As in the QPSK case, the SER is expected to degrade with increase in condition number. The degradation should be more drastic for lower bit precisions than higher ones (as observed). The SER stays stable and close to perfect with 28-bit precision even when the condition number goes above 10^4 for both signal variances. Therefore, if a system is to be designed using 16QAM, then 24-bit or 28-bit precisions should be used in order to get reliable behavior at higher condition numbers.

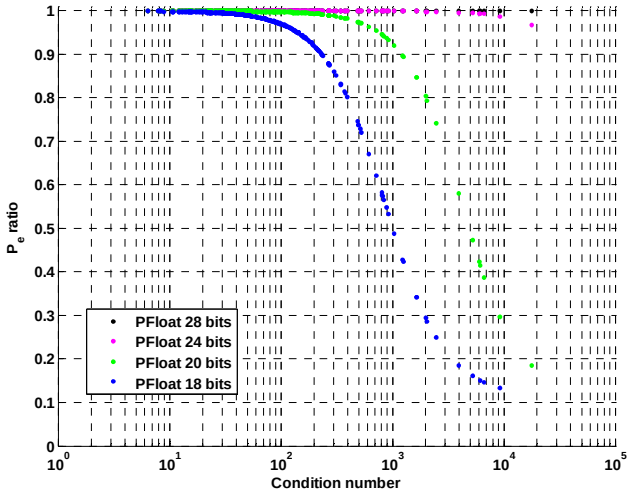

Figure 5.49: 16QAM Pe/Pe' for SNR_{line}= 50dB and Signal Variance = 2^{22}

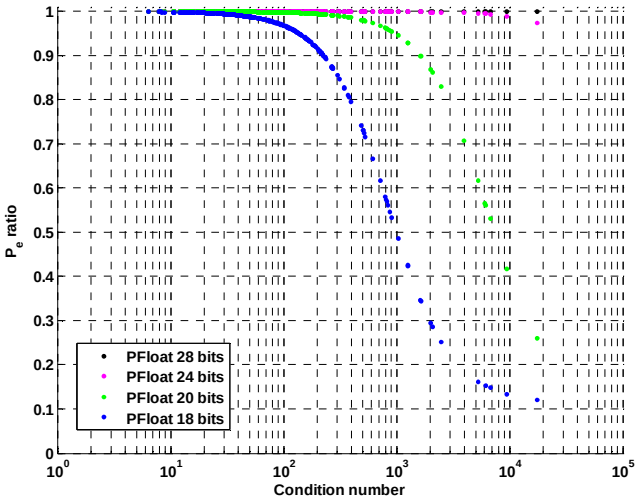

Figure 5.50: 16QAM Pe/Pe' for SNR_{line}= 50dB and Signal Variance = 2^{24}

5.3.4. 64QAM SER Loss

Figure 5.51 and Figure 5.52 show loss in SER for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $24dB$ for a 64QAM system. Similar to the 16QAM case, these plots do not show a nearly perfect ratio of $P_{sym}/P'_{sym} = 1.0$ at all bit precisions. There is a noticeable difference in the P_{sym}/P'_{sym} ratio across the tested bit precisions at higher condition numbers. No zooming is required here, as the loss in SER is clearly visible for lower precisions at the current scale. No loss in SER is observed for 28-bit precision in any of the figures shown below. Also, we notice that the maximum loss in SER is no longer small for 18-bit precision with increasing condition numbers. It is further observed that loss in SER for $\sigma_S^2 = 2^{22}$ than $\sigma_S^2 = 2^{24}$ for 18-bit precision is no longer distinguishable. This is due the fact that the noise introduced inherently by the

64QAM system is comparable to the signal variance, hence balancing out the effects of change in signal variance.

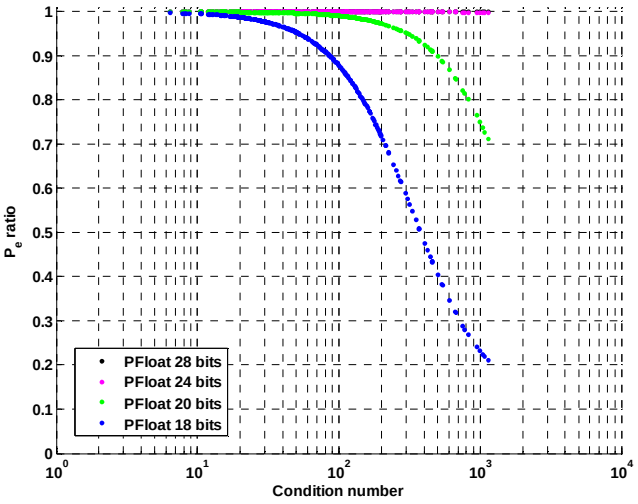

Figure 5.51: 64QAM Pe/Pe' for SNR_{line}= 24dB and Signal Variance = 2^{22}

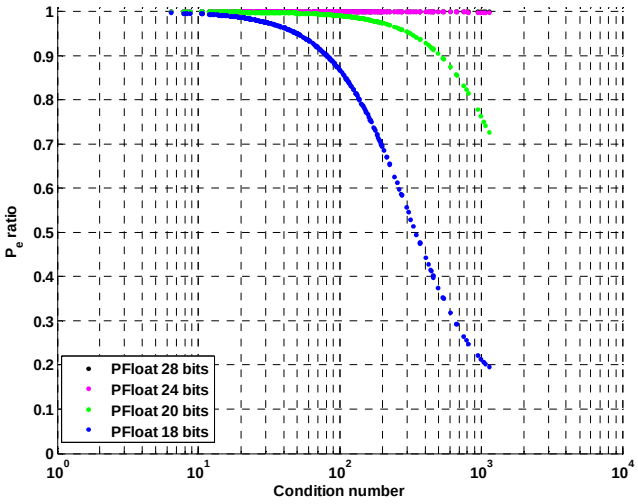

Figure 5.52: 64QAM Pe/Pe' for SNR_{line}= 24dB and Signal Variance = 2^{24}

Figures 5.53 and 5.54 show loss in SER for signal variances (σ_S^2) 2^{22} and 2^{24} respectively when SNR_{line} is fixed at $50dB$ for a 64QAM system. These plots show a clear degradation of the ratio P_{sym}/P'_{sym} at all bit precisions. Similar to the 16QAM system, the degradation for 28-bits is very small. Loss in SER is less than 0.8 for signal variance of 2^{22}, which should be acceptable in most systems. There is a significant loss in SER for both 18-bits and 20-bits, and this degradation starts very early at small condition numbers. This makes it difficult to have a system run at 18-bit or 20-bit precisions if higher input condition numbers are expected. Similar to the 16QAM case, loss in SER for 24-bits starts at a later condition number, and does not degrade very quickly. Minor loss in SER is observed for 28-bits at very high condition numbers, this makes 28-bits the best choice for a reliable 64QAM system. These results show that 24-bits is again the cut-off

point for a 64QAM system, if we want to get reasonable reliability at higher condition numbers. We observe that SER levels off at 0.1. This can mathematically verified. As condition number increases P'_{sym} also increases to a maximum value of 1. Since P_{sym} is fixed at 0.1, therefore the ratio flattens out at 0.1. If P_{sym} if fixed at a different number, then the P_{sym}/P'_{sym} ratio curve will level off at that value. Given the above observations we can conclude that if a system is to be designed using 64QAM constellations, then 24-bit or 28-bit precisions should be used in order to get reliable behavior at higher condition numbers.

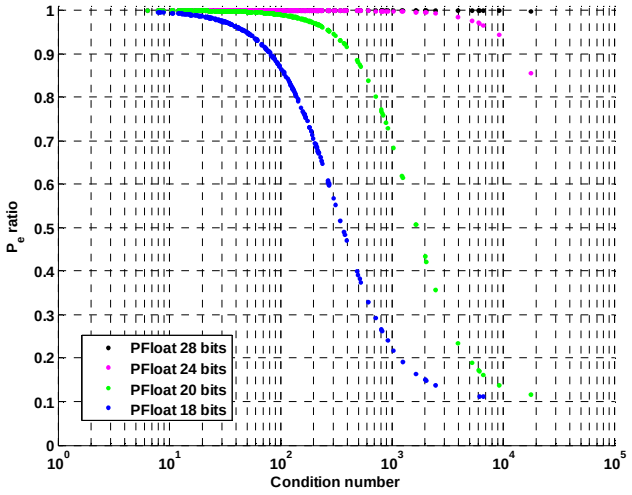

Figure 5.53: 64QAM Pe/Pe' for SNR_{line}= 50dB and Signal Variance = 2^{22}

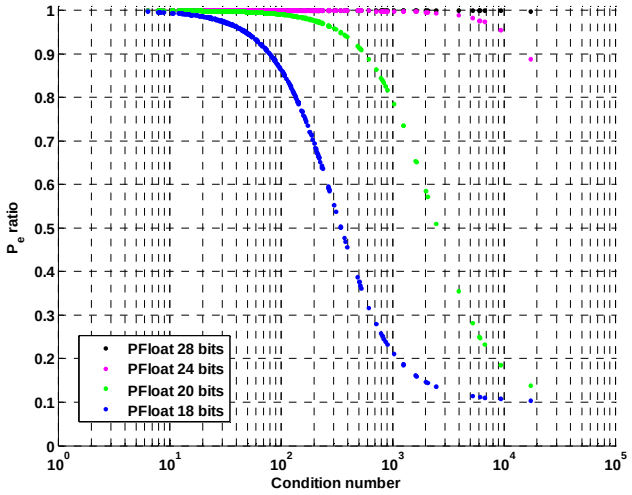

Figure 5.54: 64QAM Pe/Pe' for SNR_{line}= 50dB and Signal Variance = 2^{24}

Chapter 6 VEX Hardware Simulator

6.1. The VEX System

VEX stands for VLIW Example. “VEX includes three basic components:

1. **The VEX Instruction Set Architecture.** VEX defines a 32-bit clustered VLIW ISA that is scalable and customizable to individual application domains. The VEX ISA is loosely modeled upon the ISA of the HP/STMicroelectronics Lx/ST200 family of VLIW embedded cores. Scalability includes the ability to change the number of clusters, execution units, registers and latencies; customizability enables users to define *special-purpose* instructions in a structured way.
2. **The VEX C Compiler.** The VEX C compiler is a derivation of the Lx/ST200 C compiler, itself a descendant of the *Multiflow* C compiler. It is a robust, ISO/C89 compiler that uses *Trace Scheduling* as its global scheduling engine. A very flexible table-like machine model determines the target architecture. VEX selectively exposes some of the parameters to allow architecture exploration by changing the number of clusters, execution units, issue width and operation latencies, without having to recompile the compiler.
3. **The VEX Simulation System.** The VEX simulator is an architecture-level (functional) simulator the uses *compiled simulator* technology to achieve a speed of many equivalent ‘MIPS’. The simulation system also comes with a fairly complete set of POSIX-like *libc* and *libm* libraries (based on the GNU *newlib* libraries), a

simple built-in cache simulator (level-1 cache only), and an API that enables other plug-ins used for modeling the memory system." [24]

6.2 VEX Simulation Setup

VEX is set up to simulate the Texas Instruments C64x line of fixed point DSPs. The following table compares the TI DSP with the VEX setup. The only difference between the TI DSP and the VEX setup is that in VEX the number of ALUs cannot be odd. Therefore VEX had to be setup with four ALU units. Furthermore, VEX simulator does not allow L2 caches. Thus, two data caches are defined as L1 caches.

Table 6.1: Comparison of TI C64x DSP Specifications and VEX Simulator Setup

TI C64x DSP Specifications	VEX Simulator Setup
• VLIW clusters: 2	• VLIW clusters: 2
• Issue Width: 4	• Issue Width: 4
• ALUs per cluster: 3	• ALUs per cluster: 4
• Multipliers per cluster: 1	• Multipliers per cluster: 1
• Simultaneous Loads: 2	• Simultaneous Loads: 2
• General Register size: 64-bits	• General Register size: 64-bits
• Split L1 cache for Data and Instructions	• L1 instruction cache
• L2 cache	• 2 x L1 data caches

Following compilation flags are used to introduce timing and cycle instrumentation:

-c -ms -c99inline -fmm=config.mm -mas_G -mas_t -O4 -width 2

Details of these flags are given below:

-c Suppress the loading phase of the compilation; do not delete the '.o' files produced by the assembler. These files may be loaded by cc or ld at a later time. [24]

-ms Compile the named programs, leave the assempler-language output on corresponding files suffixed '.s' and continue to generate '.o' files normally. By default cc deletes any '.s' files that it creates. [24]

-c99inline Allow c99-style *inline* keywords to manually control inline expansion. [24]

-fmm Read machine description parameters (latency, resources, etc.) from the files specified. [24]

-mas_G Turn on gprof-style collection of profiling data. The gprof tool produces an execution profile of programs where the effect of called routines is incorporated into the profile of each caller. Profile data is taken from the call graph profile file (gmon.out by default). [24]

-mas_t Enables the collection of 'Compiled Simulator' runtime statistics, I-cache simulation and D-cache simulation when program is executed. [24]

-O4 Compiler applies level 4 optimization resulting in heavy loop unrolling. [24]

-width n Changes the number of clusters to n, must be either 1, 2 or 4. [24]

Custom instructions are defined for pseudo float add, subtract, and multiply, which are executed in VEX as standalone hardware instructions. Figure 6.1 shows the bit layout of a pseudo floating point register as customized in VEX. Bit shifting and masking is used to extract the required component as needed.

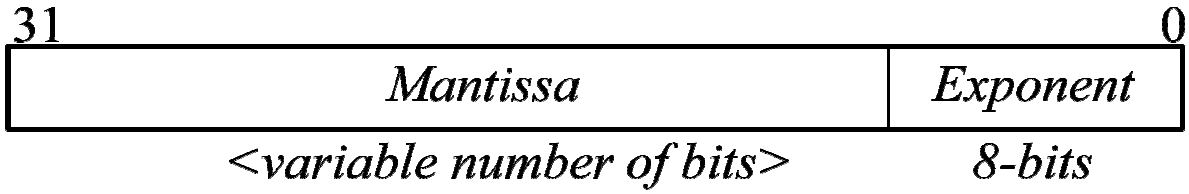

Figure 6.1: Pseudo Floating point bit allocation in VEX

6.3 VEX Simulation Results

Primary goal of doing hardware simulations with VEX is to find out the percentage of pseudo floating points operations and cycles during the total Cholesky decomposition execution. Using the profiling flags mentioned above, the following instruction and cycle numbers are obtained:

- **Total Instructions for Cholesky Decomposition:** 3345

  Instructions for Pseudo Floating point operations: 1916

  Percentage of Pseudo Floating point Instructions: 57.28%

- **Total cycles for Cholesky Decomposition:** 7555

  Cycles for Pseudo Floating point operations: 3850

  Percentage of Pseudo Floating point cycles: 50.95%

From the numbers given above it is clear that custom bit precisions in pseudo floating point hold great potential in optimizing energy and power of the overall Cholesky decomposition process.

Chapter 7 Conclusion

7.1 Conclusion

In our study we explored many bit precisions across multiple constellations that can be used to implement a MIMO system. Matrix inversion is a key part of any MIMO receiver, as the equation $Ax = b$ has to be solved every time a new channel matrix is received. Cholesky decomposition is commonly used to calculate the inverse of the channel matrix. Although LU and QR decompositions are also used in certain systems, Cholesky decomposition has the least runtime complexity and the highest stability of the three.

The primary goal of this exploration study is to determine the lowest possible bit precision for a MIMO receiver system while keeping the SNR and SER ratio within system specifications. By using less bits in calculating the inverse of the channel matrix, power, area, and energy costs can be minimized. Cholesky decomposition generates a lower triangular matrix L. This matrix is then used to calculate L^{-1} by using back-substitution. There is minimum error introduced in generating L^{-1} as no divisions take place during the calculation process. Subsequently calculations of A^{-1} matrix and the final vector are very stable as well. We explored SNR levels for various bit precisions for L, L^{-1}, and the final vector using them as checkpoints. These checkpoints help guide the hardware designer in the correct direction during the design process. At each stage the level of SNR degradation is clearly indicated and inferences are made according to the observations. Finally, loss in Symbol Error Rate (SER) is calculated by computing the

ratio P_{sym}/P'_{sym}. This is the primary metric for a system's performance. This ratio should be used as the first step to determine the bit precision for the system. After that, results for L, L^{-1}, and the result vector should be used to guide the design in the correct direction while keeping the SNR within system specifications. The VEX hardware simulations show that a significant proportion (over 50%) of instructions and cycles in Cholesky decomposition are pseudo floating point. Thus, optimizing the internal bit precision of the calculation can lead to significant power savings.

7.2 Future Prospects

There are many opportunities to explore the relationship of SNR and system bit precision in wireless communication systems. Wireless systems and digital signal processing applications commonly use SVD, and QR decomposition to calculate pseudo inverse of a channel matrix. Any wireless communication system can be optimized in power and area by using just the number of bits necessary to get the correct result. Various bit precisions can be used for different sections of the system, thus minimizing overall power and area. Another prospect is to build wireless systems where each component has adaptable internal precision. This way overall power can be reduced without compromising the minimum requirements of the wireless system.

References

[1] *LTE – an introduction*. Ericsson. 2009.

[2] D. Garrett, L. Davis, S. Brink, B. Hochwald, and G. Khagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE Journal of Solid-State Circuits*, Vol. 39, no. 9, pp. 1544-1552, Sep. 2004.

[3] G. H. Golub, and C. F. Van Loan, *Matrix Computations (Third Ed.)*. Baltimore: The John Hopkins University Press, 1996.

[4] D. S. Watkins, *Fundamentals of Matrix Computations (Second Ed.)*. New York: John Wiley & Sons, Inc., 2002.

[5] V. Kühn, *Wireless Communications over MIMO Channels*. England: John Wiley & Sons Ltd., 2006.

[6] S. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE J. Select. Areas Commun.*, vol. 16, no. 8, pp. 1451–1458, Oct. 1998.

[7] V. Tarokh, H. Jafarkhaniand, and A. R. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Transaction on Information Theory*, vol. 45, no. 5, pp. 1456–1467, July 1999.

[8] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-blast: An architecture for realizing very high data rates over rich-scattering wireless channel," *Proceedings of International Symposium on Signals, Systems, and Electronics (ISSSE)*, Pisa, Italy, pp. 295–300, Sept. 1998.

[9] Z. Khan, T. Arslan, J. S. Thompson, and A. T. Erdogan, "Dual strategy based VLSI architecture for computing pseudo inverse of channel matrix in a MIMO wireless system," in *Proceeding of IEEE International Symposium on VLSI*, pp. 12-17, 2006.

[10] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing (Second Ed.)*. New York: Cambridge University Press, 1992.

[11] C. K. Singh, N. Al-Dhahir, and P. T. Balsara, "Effect of Word-length Precision on the Performance of MIMO Systems," *IEEE International Symposium on Circuits and Systems*, pp. 2598-2601, 2007.

[12] R. Bohnke, D. Wubben, V. Kuhn, and K. D. Kammeyer, "Reduced complexity MMSE detection for BLAST architectures," *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 4, pp. 2258-2262, Dec. 2003.

[13] *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Std. 754-1985, Aug. 1985.

[14] Hai-Nam Nguyen, D. Menard, and O. Sentiey, "Dynamic Precision scaling for low power WCDMA receiver," *IEEE International Symposium on Circuits and Systems*, pp. 205-208, May 2009.

[15] D. Menard, R. Serizel, R. Rocher, and O. Sentieys, "Accuracy Constraint Determination in Fixed-Point System Design," *EURASIP Journal on Embedded Systems*, vol. 2008.

[16] Z. Khan, T. Arslan, J. S. Thompson, and A. T. Erdogan, "Analysis and Implementation of Multiple-Input, Multiple-Output VBLAST Receiver From Area and Power Efficiency Perspective," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, no. 11, pp. 1281-1286, Nov. 2006.

[17] H. Kim, C. B. Chae, G. Veciana, and R. W. Heath, "Energy-Efficient Adaptive MIMO Systems Leveraging Dynamic Spare Capacity," *42nd Annual Conference on Information Sciences and Systems (CISS)*, pp. 68-73, 2008.

[18] W. W. L. Ho, and Y. C. Liang, "Efficient Resource Allocation for Power Minimization in MIMO-OFDM Downlink," *IEEE 68th Vehicular Technology Conference (VTC 2008-Fall)*, pp. 1-5, 2008.

[19] A. Irturk, B. Benson, N. Laptev, and R. Kastner, "Architectural Optimization of Decomposition Algorithms for Wireless Communication Systems," *IEEE Wireless Communications and Networking Conference (WCNC)*, April 2009.

[20] A. Irturk, B. Benson, A. Arfaee, and R. Kastner, "Automatic Generation of Decomposition based Matrix Inversion Architectures", *IEEE International Conference on Field-Programmable Technology (ICFPT)*, Dec. 2008.

[21] A. Irturk, B. Benson, S. Mirzaei, and R. Kastner, "GUSTO: An Automatic Generation and Optimization Tool for Matrix Inversion Architectures," *ACM Transactions on Embedded Computing Systems*, to appear.

[22] A. Hosangadi, F. Fallah, R. Kastner, “Algebraic Methods for Optimizing Constant Multiplications in Linear Systems,” *Springer Journal of VLSI Signal Processing*, vol. 49, issue 1, pp. 31-50, Oct. 2007.

[23] A. Arfaee, A. Irturk, F. Fallah, and R. Kastner, “Xquasher: A Tool for Efficient Computation of Multiple Linear Expressions,” *Design Automation Conference (DAC)*, July 2009.

[24] *The Vex System*, Hewlett Packard Labs, 2009.