A NOVEL APPROACH TO MODULATION CLASSIFICATION IN
COGNITIVE RADIOS

A Dissertation in
Electrical Engineering
by
Okhtay Azarmanesh

© 2011 Okhtay Azarmanesh

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2011
The thesis of Okhtay Azarmanesh was reviewed and approved* by the following:

Sven G. Bilén  
Associate Professor of Engineering Design, Electrical Engineering, and Aerospace Engineering  
Dissertation Advisor, Chair of Committee

Julio V. Urbina  
Assistant Professor of Electrical Engineering

James K. Breakall  
Professor of Electrical Engineering

John F. Doherty  
Professor of Electrical Engineering

John D. Mitchell  
Professor of Electrical Engineering

Kyusun Choi  
Assistant Professor of Computer Science and Engineering

Kultegin Aydin  
Professor of Electrical Engineering  
Department Head

*Signatures are on file in the Graduate School.
Abstract

Thesis Statement: In cognitive radios, Blind Modulation Classification is an important intermediate step between signal detection and demodulation. There has been an increasing need and thus search for optimal modulation classifiers due to ever increasing variety of digital modulations.

The modulation classification technique discussed here is being designed for a real-time Software-Defined Radio (SDR) system to be implemented on SDR development boards and it is robust and efficient with a processing time overhead low enough to allow the software radio to maintain its real-time operating objectives. We are investigating classification of digital single-carrier modulations as well as multi-carrier modulations. The method is to use the waveform’s In-phase–Quadrature (I–Q) diagrams and, by employing clustering algorithms on them, determine the type of modulation being transmitted.

For classifying single-/multi-carrier modulations, we further study existing methods to find the appropriate Gaussianity test to classify single-carrier signals from multi-carrier ones. This technique may require a lot of processing power; however, with today’s technology, it is feasible. Thus, we try to find the test with the best error rate and least amount of processing time.

We also include this algorithm with methods to extract the features of an Orthogonal Frequency Division Multiplexing (OFDM) signal in the case of multi-carrier modulations. We also utilize the clustering algorithms to propose a new method in correcting frequency imbalances and I–Q offsets in OFDM signals as a mean of signal synchronization.

This new Modulation Classification method will be capable of determining the type of modulation scheme among different PAM, PSK, QAM, and OFDM modulations and can be further expanded to include any new modulation scheme.
Table of Contents

List of Figures viii
List of Tables xv
List of Symbols xvi
Acknowledgments xix

Chapter 1
Introduction 1
1.1 Motivation 1
1.2 Basics of Software-Defined Radios and Cognitive Radios 2
1.2.1 Software-Defined Radios 3
1.2.2 Cognitive Radios 4
1.3 Contributions of This Work 9
1.4 Outline of the Thesis 11

Chapter 2
Background 13
2.1 Introduction 13
2.2 Modulation Classification: Problem Formulation 13
2.2.1 Assumptions Concerning SNR, Sampling Rate, and Time and Carrier Recovery 16
2.3 Background: Existing Methods 17
2.3.1 General MC Algorithms 17
2.3.2 Gaussianity Tests 23
2.3.3 Image Processing and Pattern Recognition Techniques 24
2.3.4 OFDM Feature Extraction 25
Chapter 3

Modulation of Single-Carrier Signals vs. Multi-Carrier Signals

3.1 Introduction

3.2 Comparison of Modulations for the Purpose of Classification

3.2.1 $M$-PSK ($M$-ary Phase Shift Keying)

3.2.2 $M$-QAM ($M$-ary Quadrature Amplitude Modulation)

3.2.3 PAM (Pulse Amplitude Modulation)

3.2.4 OFDM (Orthogonal Frequency Division Multiplexing)

3.2.5 Continuous Phase Modulations (CPM)

3.2.6 $M$-FSK ($M$-ary Frequency Shift Keying)

3.3 Gaussianity Tests as Means of Classification

3.3.1 $\chi^2$ Test

3.3.2 Jarque–Bera Test

3.3.3 Giannakis–Tsatsanis Test

3.3.4 Kolmogorov–Smirnov Test

3.3.5 Anderson–Darling Test

3.3.6 D'Agostino–Pearson Test

3.3.7 Shapiro–Wilk Test

3.3.8 Cramer–von Mises Test

3.3.9 Lilliefors Test

3.4 Single-/Multi-Carrier MC Simulations

3.5 Summary

Chapter 4

New Method for Modulation Classification — Overview of the Method

4.1 Single-Carrier Modulation Classification

4.1.1 Clustering Algorithms

4.1.1.1 $k$-means

4.1.1.2 $k$-center

4.1.2 Issues With Single-Carrier Modulation Classification

4.2 Multi-Carrier Modulation Classification

4.2.1 Cyclostationarity Test for Checking Validity of Gaussianity Tests

4.2.2 Estimation of Cyclic Prefix Duration

4.2.3 Estimation of Number of Subcarriers
4.2.4 OFDM Synchronization ........................................... 89
4.2.5 Sensitivity to Phase Noise ....................................... 90
4.2.6 Sensitivity to Frequency Offset ................................. 91
4.2.7 Sensitivity to Timing Errors ..................................... 92
4.2.8 Using Clustering Algorithm for Correcting Offsets in OFDM
    Signal ..................................................................... 93
4.3 Summary ..................................................................... 94

Chapter 5
Performance Test for New MC Method 104
5.1 Introduction ............................................................. 104
5.2 Using Simulink for Hardware Implementation ............... 104
5.3 Clustering Algorithms in I–Q Diagrams ......................... 105
5.4 On the Complexity of the Algorithm ......................... 137
5.5 Summary ................................................................. 139

Chapter 6
Conclusions and Future Work 141
6.1 Introduction ............................................................. 141
6.2 Conclusions .............................................................. 141
6.3 Future Research ......................................................... 143
   6.3.1 Automatic Constellation Detection ..................... 143
6.3.2 Optimization of Gaussianity Tests ......................... 145
6.3.3 Protocol Analysis ................................................ 145
6.3.4 Hardware Implementation .................................... 145
   6.3.4.1 Lyrtech SignalWAVe .................................... 146
   6.3.4.2 Universal Software Radio Peripheral .............. 146
6.3.5 Transmission Channels ........................................ 147
   6.3.5.1 Additive White Gaussian Noise (AWGN) Channels 149
   6.3.5.2 Rayleigh Fading Channels .............................. 149
   6.3.5.3 Rician Fading Channels ................................. 149

Appendix A
Deriving Gaussianity using Kurtosis and Skewness of a Distribu-

Appendix B
The Giannakis–Tsatsanis Test: Derivation of Formulas 154
Appendix C
Cluster-finding Algorithms in Pattern Recognition 157
C.1 $H$-means .................................................. 157
C.2 $k$-means .................................................... 158
C.3 $k$-center .................................................. 158
C.4 $k$-medians ............................................... 158
C.5 $k$-medians Squared .................................. 159
C.6 Fuzzy $c$-means ........................................ 159
C.7 QT Clustering ............................................ 159

Appendix D
Glossary 161

Appendix E
MATLAB Codes 163
 E.1 Gaussianity Tests ........................................ 163
     E.1.1 Giannakis–Tsatsanis .................................. 163
     E.1.2 Anderson–Darling .................................. 164
 E.2 Gaussianity Test Simulation Code .................. 164
 E.3 Clustering Algorithms ................................. 166
     E.3.1 Main Code ........................................ 166
     E.3.2 $k$-center Function ............................... 170
 E.4 OFDM Simulations ................................. 171
     E.4.1 Clustering Algorithms for OFDM Signals - Main Code  171
     E.4.2 Clustering Algorithms for OFDM Signals - Function  172
 E.5 BPSK vs. QPSK Classifier for Comparing with Other Classifiers  175
     E.5.1 Main Code ........................................ 175
     E.5.2 Classifier Function ............................... 175

Bibliography 177
List of Figures

1  Cedalion standing on the shoulders of Orion, Nicolas Poussin (1594–1665), Metropolitan Museum of Art. .......................... xix

1.1  This figure shows the different steps of radio evolution from HR (Hardware Radio) to SCR (Software Controlled Radio), SDR (Software Defined Radio), ISR (Ideal Software Radio), and USR (Ultimate Software Radio). .......................................................... 4

1.2  The general behavior of a cognitive radio and where the modulation classification is used. ..................................................... 10

1.3  Detailed steps in the proposed algorithm for modulation classification. The blocks in purple include the contributions of this research. 11

2.1  Automatic modulation classification for choosing the correct demodulation method. ......................................................... 16

3.1  Tree structure for proposed modulation classification. .......... 28

3.2  I–Q diagram of QAM modulations: (a) 4-QAM, (b) 16-QAM, (c) 64-QAM. ................................................................. 31

3.3  I–Q diagram of PSK modulations: (a) 4-PSK, (b) 16-PSK, (c) 64-PSK. ................................................................. 32

3.4  I–Q diagram of PAM modulations: (a) 4-PAM, (b) 16-PAM, (c) 64-PAM. ................................................................. 32

3.5  I–Q diagram of a MSK modulation. ....................................... 33

3.6  I–Q diagram of FSK modulations: (a) 4-FSK, (b) 16-FSK, (c) 64-FSK. ................................................................. 34

3.7  Comparing $d_{G,4}$ from the first and second implementations of Giannakis–Tsatsanis test for a randomly generated sequence. ........ 40

3.8  $d_{G,4}$ vs. SNR, from the first implementation of Giannakis–Tsatsanis test for a randomly generated sequence: (a) raw data, (b) normalized data. ................................................................. 42
3.9 $d_{G,4}$ vs. SNR, from the second implementation of Giannakis–Tsatsanis test for a randomly generated sequence: (a) raw data, (b) normalized data. ........................................................................................................ 43

3.10 $d_{G,4}$ vs. SNR, from the first implementation of Giannakis–Tsatsanis test for a QPSK signal passing through AWGN channel: (a) raw data, (b) normalized data. .......................................................... 44

3.11 $d_{G,4}$ vs. SNR, from the second implementation of Giannakis–Tsatsanis test for a QPSK signal passing through AWGN channel: (a) raw data, (b) normalized data. .......................................................... 45

3.12 Cramer–von Mises statistic vs. SNR for a QAM signal. ......................................................... 49

3.13 Error rate vs. SNR for Gaussianity tests, when dealing with a Gaussian signal (Gaussian noise or multi-carrier signal). ........................................................................................................ 53

3.14 Error rate vs. SNR for Gaussianity tests, when dealing with a uniformly distributed random signal that has passed through AWGN channel. .................................................................................. 54

3.15 Error rate vs. SNR for Gaussianity tests, for the second scenario, when a randomly generated signal passes through AWGN channel. .................................................................................. 55

3.16 Error rate vs. SNR for Gaussianity tests, for the third scenario, when a QPSK signal passes through AWGN channel. ........................................................................................................ 56


4.1 Difference in the performance of $k$-means algorithm for 4-QAM and 16-QAM modulations, [(a), (b)] when initialized with $k$-center algorithm, and [(c),(d)] when initialized with uniformly distributed random samples. The simulated signal has 256 samples with SNR = 7 dB. 

4.2 Difference in the performance of $k$-center algorithm for 4-QAM and 16-QAM modulations, [(a), (b)] when used to initialize the $k$-means algorithm, and [(c),(d)] when used independently. The simulated signal has 256 samples with SNR = 5 dB. 

4.3 Comparison of error rate for clustering algorithm with $k$-center initialization vs. uniformly distributed random initialization, for SNR = 5 dB. 

4.4 Comparison of error rate for clustering algorithm with $k$-center initialization vs. uniformly distributed random initialization, for SNR = 15 dB. 

4.5 Comparison of error rate for clustering algorithm with $k$-center initialization vs. uniformly distributed random initialization, for SNR = 30 dB. 

4.6 Clustering algorithm performed on (a) 4-QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM signals with SNR = 30 dB and 512 samples. The red dots show the center of each cluster. 

4.7 Clustering algorithm performed on (a) 4-QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM signals with SNR = 30 dB and 1024 samples. The red dots show the center of each cluster. 

4.8 Clustering algorithm performed on (a) QPSK, (b) 8-PSK, (c) 16-PSK, and (d) 32-PSK signals with SNR = 30 dB and 512 samples. The red dots show the center of each cluster. 

4.9 Clustering algorithm performed on (a) 4-QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM signals with SNR = 20 dB and 512 samples. The red dots show the center of each cluster. 

4.10 Clustering algorithm performed on (a) QPSK, (b) 8-PSK, (c) 16-PSK, and (d) 32-PSK signals with SNR = 20 dB and 512 samples. The red dots show the center of each cluster.
4.11 Clustering algorithm performed on (a) 4-QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM signals with SNR = 10 dB and 512 samples. The red dots show the center of each cluster.

4.12 Clustering algorithm performed on (a) QPSK, (b) 8-PSK, (c) 16-PSK, and (d) 32-PSK signals with SNR = 10 dB and 512 samples. The red dots show the center of each cluster.

4.13 Clustering algorithm performed on (a) 4-QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM signals with SNR = 5 dB and 512 samples. The red dots show the center of each cluster.

4.14 Clustering algorithm performed on (a) QPSK, (b) 8-PSK, (c) 16-PSK, and (d) 32-PSK signals with SNR = 5 dB and 512 samples. The red dots show the center of each cluster.

4.15 Clustering algorithm performed on (a) 2-PAM, (b) 4-PAM, (c) 8-PAM, and (d) 16-PAM signals with SNR = 20 dB and 512 samples. The red dots show the center of each cluster.

4.16 Cyclic prefix being copied for eliminating ISI.

4.17 The result of autocorrelation test. The peak shows where the cyclic prefix ends and the data begins in the duration of one symbol.

4.18 Phase sensitivity on SNR degradation for 64-QAM ($E_s/N_0 = 19$ dB), 16-QAM ($E_s/N_0 = 14.5$ dB), QPSK ($E_s/N_0 = 10.5$ dB).

4.19 Measuring the angle to calculate the I–Q phase imbalance in a 16-QAM modulation.

4.20 Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 5% carrier frequency offset, (c) with 5% frame time offset, and (d) with 5% sampling clock offset. The red stars show the center of each cluster.

4.21 Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 10% carrier frequency offset, (c) with 10% frame time offset, and (d) with 10% sampling clock offset. The red stars show the center of each cluster.

4.22 Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 15% carrier frequency offset, (c) with 15% frame time offset, and (d) with 15% sampling clock offset. The red stars show the center of each cluster.

4.23 Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 10% I–Q gain imbalance, (c) with 10% I–Q phase imbalance, and (d) with 10% I–Q differential time. The red stars show the center of each cluster.
4.24 Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 20% I–Q gain imbalance, (c) with 20% I–Q phase imbalance, and (d) with 20% I–Q differential time. The red stars show the center of each cluster. .................................................. 100

4.25 Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 30% I–Q gain imbalance, (c) with 30% I–Q phase imbalance, and (d) with 30% I–Q differential time. The red stars show the center of each cluster. .................................................. 101

4.26 Correcting the frame time offset in an OFDM signal (16-QAM): measured offset by classifying algorithm versus actual offset. ................. 102

4.27 Correcting the I–Q phase imbalance in an OFDM signal (16-QAM): measured imbalance by calculating the angle between the axis of 16-QAM signal with x axis using classifying algorithm. The unit on the x axis shows the percentage of the I–Q phase imbalance present. 103

5.1 Simulink model of a simplified OFDM transmitter/receiver. ............ 106
5.2 (a) Spectrum of transmitted signal and (b) spectrum of received signal. .................................................................................. 107
5.3 I–Q diagrams of (a) transmitted and (b) received signals for OFDM–16QAM modulation. .................................................. 107
5.4 Relative error with respect to number of samples in a 4-QAM modulation with SNR = 20 dB. .................................................. 108
5.5 Relative error with respect to number of samples in a 8-QAM modulation with SNR = 20 dB. .................................................. 109
5.6 Relative error with respect to number of samples in a 16-QAM modulation with SNR = 20 dB. .............................. 110
5.7 Relative error with respect to number of samples in a 64-QAM modulation with SNR = 20 dB. .................................................. 111
5.8 Relative error with respect to number of samples in a 8-PSK modulation with SNR = 20 dB. .................................................. 112
5.9 Relative error with respect to number of samples in a 16-PSK modulation with SNR = 20 dB. .................................................. 113
5.10 Relative error with respect to number of bits in a 4-QAM modulation with SNR = 20 dB. .................................................. 114
5.11 Relative error with respect to number of bits in a 8-QAM modulation with SNR = 20 dB. .................................................. 115
5.12 Relative error with respect to number of bits in a 64-QAM modulation with SNR = 20 dB. .................................................. 116
5.13 Relative error with respect to number of bits in a 8-PSK modulation with SNR = 20 dB. .................................................. 117
5.14 Relative error shows the deviation of found cluster centers from the actual cluster centers in 4-QAM and 8-QAM signals with respect to SNR. .......................... 118
5.15 Relative error shows the deviation of found cluster centers from the actual cluster centers in 8-PSK signal with respect to SNR. .......................... 119
5.16 Comparison of relative error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 20 dB. .......................... 120
5.17 Comparison of relative error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 10 dB. .......................... 121
5.18 Comparison of relative error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 5 dB. .......................... 122
5.19 Comparison of relative error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 3 dB. .......................... 123
5.20 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 20 dB. .......................... 124
5.21 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 10 dB. .......................... 125
5.22 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 5 dB. .......................... 126
5.23 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 3 dB. .......................... 127
5.24 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 16-PSK modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 20 dB. .......................... 128
5.25 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 16-PSK modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 3 dB. .......................... 129
5.26 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 16-QAM modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 20 dB. .......................... 130
5.27 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 16-QAM modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 3 dB. ............................................. 131
5.28 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 64-QAM modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 20 dB. ................................. 132
5.29 Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 64-QAM modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 3 dB. ................................. 133

6.1 Distance between symbols in I–Q diagram as a way of determining the type of modulation. (a) QPSK, (b) 8-PSK, (c) 16-QAM, and (d) a type of 8-QAM. Arrows indicate the neighboring symbols. 144
6.2 USRP and GNU Radio testbed configuration .......................... 147
6.3 User interface for choosing the desired Gaussianity test, written in MATLAB ......................................................... 148
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Feature-based classifiers</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>Likelihood-based classifiers</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>Normalization factors</td>
<td>30</td>
</tr>
<tr>
<td>3.2</td>
<td>Percentage points for Cramer–von Mises test of Gaussianity with unknown $\mu$ and $\sigma^2$.</td>
<td>48</td>
</tr>
<tr>
<td>5.1</td>
<td>Classifier results for 16-QAM modulated signal</td>
<td>134</td>
</tr>
<tr>
<td>5.2</td>
<td>Classifier results for 64-QAM modulated signal</td>
<td>134</td>
</tr>
<tr>
<td>5.3</td>
<td>Classifier results for 8-PSK modulated signal</td>
<td>134</td>
</tr>
<tr>
<td>5.4</td>
<td>Classifier results for 16-PSK modulated signal</td>
<td>135</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparing with previous methods</td>
<td>136</td>
</tr>
<tr>
<td>5.6</td>
<td>Comparing with previous methods</td>
<td>137</td>
</tr>
</tbody>
</table>
List of Symbols

\((\cdot)^*\)  complex conjugate of a matrix or a vector

\(\|p - q\|\)  Euclidean distance of vectors \(p\) and \(q\)

\(B\)  channel bandwidth in hertz

\(\mathbb{C}\)  set of complex numbers

\(D_{\text{deg}}\)  total degradation

\(D_f\)  frequency degradation

\(D_p\)  phase degradation

\(\mathbb{E}\{\cdot\}\)  expected value operator

\(E_b\)  signal energy per bit

\(E_s\)  baseband signal energy

\(F(\cdot)\)  cumulative distribution function (CDF)

\(F_N(\cdot)\)  empirical distribution function (empirical CDF) with \(N\) data points

\(\mathcal{H}_0\)  null hypothesis

\(\mathcal{H}_1\)  alternative hypothesis

\(I(t)\)  in-phase component of a signal

\(\Im(\cdot)\)  imaginary part of a signal

\(N_0\)  noise power spectral density
$O$  big O notation for indicating the upper bound on the growth rate of a function

$Q(t)$  quadrature component of a signal

$\mathbb{R}^d$  $d$ dimensional vector space

$\mathbb{R}(\cdot)$  real part of a signal

$\hat{R}_{rr}(\cdot)$  cyclic autocorrelation

$T_{cp}$  cyclic prefix duration

$T_d$  data duration

$T_s$  symbol duration

$a$  scalars in lower case

$c$  vectors are column vectors and in lower case bold

$c_{4j}(\cdot)$  4th-order cumulants of imaginary part of a signal

$c_{4r}(\cdot)$  4th-order cumulants of real part of a signal

$\hat{c}_r^4(\cdot, \cdot, \cdot)$  4th-order cumulant of signal $r(\cdot)$

$d_{AD}$  the Anderson–Darling test statistic

$d_{CV}$  the Cramer–von Mises test statistic

$d_{DP}$  the D’Agostino–Pearson test statistic

$d_{G,A}$  the Giannakis–Tsatsanis test statistic

$d_{JB}$  the Jarque–Bera test statistic

$d_{KS}$  the Kolmogorov–Smirnov test statistic

$d_{L}$  the Lilliefors test statistic

$d_{SW}$  the Shapiro–Wilk test statistic

$d_{\chi^2}$  the $\chi^2$ test statistic

$\text{erfc}(\cdot)$  the complementary error function defined as $\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$
 probability density function (PDF)

\( f_c \)  carrier frequency

\( f_s \)  sampling frequency

\( \ln(\cdot) \)  logarithm to base \( e \)

\( \log_2(\cdot) \)  logarithm to base 2

\( \mathbf{r} \)  vector of received samples

\( r(t) \)  signal received by the receiver

\( \hat{\mathbf{r}}_{rp(\cdot)} \)  cyclic autocorrelation vector

\( \sup(\cdot) \)  supremum or the least upper bound of a set

\( x(t) \)  signal transmitted over the channel

\( \Delta f \)  carrier frequency offset

\( \Sigma_c \)  covariance matrix of vector \( \mathbf{c} \)

\( \gamma_1 \)  skewness or third standardized moment

\( \gamma_2 \)  kurtosis or fourth standardized moment

\( \varepsilon \)  error, presented in tests and simulations

\( \varepsilon_{\text{CDE}} \)  cumulative deviation error (CDE)

\( \theta_c \)  carrier phase

\( \mu \)  mean of a distribution

\( \sigma \)  standard deviation of a distribution

\( \phi(\cdot) \)  kernel function
Acknowledgments

"Dicebat Bernardus Carnotensis nos esse quasi nanos, gigantium humeris insidentes, ut possimus plura eis et remotiora videre, non utique proprii visus acumine, aut eminencia corporis, sed quia in altum subvenimur et extollimur magnitudine gigantea.

“Bernard of Chartres used to say that we are like dwarfs on the shoulders of giants, so that we can see more than they, and things at a greater distance, not by virtue of any sharpness of sight on our part, or any physical distinction, but because we are carried high and raised up by their giant size.” — The Metalogicon of John Salisbury, 1159 AD

To stand on the shoulders of giants. Although one never stops from learning, I

Figure 1. Cedalion standing on the shoulders of Orion, Nicolas Poussin (1594–1665), Metropolitan Museum of Art.
have come to an end of my formal educational life and, when I look back, I see many
giants on whose shoulders I’ve stood and who peek through time at me. This work
represents five years of my life and so many people have helped me and guided me
through it. Now I only see fit to express my humble gratitude and appreciation to
some of them by thanking them, however brief it may be.

First and foremost, I must express my deepest gratitude to my advisor Professor
Sven G. Bilén, who gave me a unique opportunity to work on Cognitive Radios and
was most supportive and understanding through all the ups and downs of the past
five years that I had the honor of having him as my advisor and whose yearning
for excellence pushed me to always perform at my best.

However, the two most important giants of my life have been my parents. By
standing on their shoulders, my father has showed me the new horizons in science
and engineering and my mother has opened my eyes to new doors of art, literature,
and history. There are no possible ways to express my gratitude for all they have
done for me.

I have also been lucky to have a number of formidable engineers in our lab
who have helped me throughout my work. Pradyumna Desale and Xiaoxiang Xiao
who were my teammates for SDR Forum’s Smart Radio Challenge and helped
me with a few ideas that I incorporated in my work, Aseem Singh who made me
think beyond my topic of research through interesting topics of discussion that he
always had with him, and finally Robert Capuro, who gave us unwavering support
whenever we needed it and specially helped me with proofreading of this thesis.

During these five years as a Ph.D. student, I have also had the opportunity
to meet more outstanding scientists and engineers that I can possibly fit in these
pages, but I feel the urge to particularly mention two of them: Professor Julio
Urbina with whom I had my first SDR course and guided me through my first
steps down this path, and the late Professor Nirmal K. Bose who taught me more
than I realized by being a great teacher and a greater human being.

But completing my Ph.D. in University Park has been more than a professional
experience for me. It has brought many life experiences that having any one of
them throughout someone’s life would make that person very lucky. How can
I forget the many friends who have come and gone during these past years and
who have left their footprints forever in my life and soul? There is no possible
way for me to name them all, but I would be so ungrateful if I didn’t name a
few. These people have been there for me during all the good times and bad
times and, without their support, I would not have made it so far. My dear
Shirin, who in the past one year has made my life brighter and has unconditionally
supported me; Jérémie, my oldest friend at Penn State, who introduced me to life
in this small Pennsylvania town; my sisters, Negar, Deniz, Claire Vaille, Amparo
Soler Martinez, Lucie Lecestre, Serena Polverigiani, Mélanie Gasnier, Clémentine
Roux, Khanh Nguyen, Mastaneh Sharafi, Samira Khalili, and Azadeh Rabbani; and my brothers Daniel Bethge, Amir Lotfi, Behdad Masih Tehrani, Sebastien De Larquier, Amirreza Ghasemi, and Ishan Behoora, who always believed in me but also offered their valuable criticism whenever I needed it; and all the others that would be worth making this section longer than the dissertation itself. For you all, as J.R.R. Tolkien has beautifully put it, “I don’t know half of you half as well as I should like; and I like less than half of you half as well as you deserve.”

Lastly, I would like to thank you, the one reading this dissertation, because your interest in my work justifies the five years that I have invested in it.

Okhtay Azarmanesh

*University Park, PA*

*2011*

“Twenty years from now you will be more disappointed by the things that you didn’t do than by the ones you did do. So throw off the bowlines. Sail away from the safe harbor. Catch the trade winds in your sails. Explore. Dream. Discover.”

— *Mark Twain*
To my sisters,
Negar and Deniz
Chapter 1

Introduction

“Cogito, egro sum. [I think, therefore I am]” —René Descartes

1.1 Motivation

In a world of increasing mobility, there is a growing need for people to communicate with each other and have timely access to information regardless of the locations of the individuals or the information.

Wireless technology is changing our social and individual lives while creating endless opportunities for new and varied services. Due to these services, some consumer devices are getting more attention than others. However, while cell phones, tablets, and laptops are receiving most of the attention, the impact of wireless technology is much broader and deeper, e.g., through sensor networks, smart grid control, home automation, medical embedded wireless devices, and entertainment systems.

In order to make these technologies possible, software-defined radios (SDR) and cognitive radios (CR) are becoming increasingly commonplace, especially in the commercial arena, and the technology to support them has become advanced enough to enable use of new modulation-classification (MC) techniques.

SDR and CR are powerful technologies for a number of major applications. They will improve interoperability among different telecommunication partners and public safety officers while solving bandwidth problems by reducing the number
and types of radios required to accomplish operational objectives.

As an example, consider a plane flying across the globe where each country’s airspace has different communication standards, frequencies, and protocols. With a cognitive radio, the flight plan is programmed into the radio, and just as GPS tells the pilot where they are, the radio would adapt to the communication architectures of the airspace throughout the flight, without the need for pilot interaction.

CR platforms are also powerful tools for compensating for inadequate infrastructure. They can enhance coverage around these gaps by being aware of the areas with poor signal quality, learning the best physical layer parameters and taking action to compensate for loss of signal by adjusting power, bandwidth, or coding, thus aiding the cellular network by informing the system of identified gaps.

CR applications have already covered vast areas of the wireless domain. Cheaper radios, spectrum trading, automated interoperability, intelligent beam-forming, improved link reliability, opportunistic spectrum utilization, collaborative techniques, and advanced networking are already being implemented in various military and commercial systems.

1.2 Basics of Software-Defined Radios and Cognitive Radios

Considering the importance of wireless communications in our daily lives, CRs are the new generation of communication devices that add artificial intelligence to their communication backbone. With an increasing number of devices becoming wireless, the finite radio spectrum is becoming more and more occupied and problems of traffic congestion and interference are becoming more frequent. The interest in developing new spectrum utilization technologies — combined with both the introduction of SDR and the realization that machine learning can be applied to radios — is creating intriguing possibilities for promising technologies that are being incorporated in CRs.

Modulation classification (MC) in particular, plays an important role in various civilian and military applications. A crowded electromagnetic spectrum is a great challenge for communication engineers when implementing advanced information
services and systems. Furthermore, blind recognition of the modulation format of the received signal is an important capability needed in military and commercial systems, especially in SDR, which copes with a variety of communication systems. MC can also be used with an intelligent receiver, yielding an increase in the throughput by reducing the overhead. This is achieved by transmitting useful data instead of sending the modulation information of the transmitted signal. Such applications have signified the need for cognitive communication systems, for which the automatic recognition of the modulation of a detected signal is a major task.

To start this discussion, we now present a few definitions that exist for SDR and CR and then we present the organization of this thesis.

### 1.2.1 Software-Defined Radios

A number of definitions exist to describe what is known as Software-Defined Radio, also known as Software Radio or SDR. The Wireless Innovations Forum with the Institute of Electrical and Electronics Engineers (IEEE) P1900.1 group have worked to establish a definition of SDR that provides consistency and a clear overview of the technology and its associated benefits [SDR Forum Base Station Working Group, 2002]:

**Definition 1.1.** *SDR is a radio in which some or all of the physical layer functions are software defined.*

To better illustrate, SDR can also be categorized as part of the evolution of radio from Hardware Radio to Ultimate Software Radio, as shown in Figure 1.1.

HR (Hardware Radio) corresponds to the classical view of radio, consisting of only hardware components that cannot be modified. SCR (Software Controlled Radio) has only its control functions defined in software. Its interconnections and power levels, for example, are defined in software, but not its modulation type and frequency band. SDR (Software-Defined Radio) provides control of variety of modulation techniques, e.g., wide band or narrow band operation, communication security functions, waveform requirements, etc. The frequency band will still be a limitation on the front-end to the antenna. ISR (Ideal Software Radio) eliminates analog amplification and mixing, which are performed prior to Analog-to-Digital
This figure shows the different steps of radio evolution from HR (Hardware Radio) to SCR (Software Controlled Radio), SDR (Software Defined Radio), ISR (Ideal Software Radio), and USR (Ultimate Software Radio).

Conversion (ADC). Software-defined parts of the radio extend to the entire system with analog conversion only at the antenna. Finally, USR (Ultimate Software Radio) is defined as the theoretical limit and, it accepts fully programmable traffic and supports wide range of frequencies, modulation types, air interfaces, and application software. Some of its capabilities include: rapid switching from one air interface to another, use of GPS and providing video through local broadcast or satellite transmission.

Based on this definition, it can be seen that SDR is an extremely multidisciplinary area, requiring a breadth and depth of knowledge and background in a wide variety of subjects.

1.2.2 Cognitive Radios

Broadly speaking, cognition means “the act of knowing or knowledge, which relates to different thought processes and actions, and how the outcome of these processes affects other groups working in close relation” [Lycan and Prinz, 2008]. Based on Britannica’s definition, cognition is closely related to concepts of the mind such as perceiving, conceiving, reasoning, and intelligence. We generally associate the word cognition with cognitive science, the study of mind and intelligence.
This definition leads to question: how can a radio network system become intelligent? Isn’t it intelligent enough? How will concepts like CR help improve our existing wireless communication networks? How will future products evolve based on the concepts of CR? How will a cognitive engine change the face of our wireless systems?

With the rapid growth of wireless applications, we are facing an ever increasing demand for more radio spectrum. We all have experienced frustration when our mobile calls are blocked because of the unavailability of free radio circuits or when the quality of the service is unacceptable. These situations are common especially when a large number of users are all sharing the frequency spectrum without giving any priority to primary users and their needs and dealing with all users the same way because of lack of intelligence; or in a completely opposite situation, all the spectrum is assigned to primary users even when they are not using it all the time. The result is inefficient use of a very limited resource. Since our existing radio networks are trained to route calls or provide radio reception depending upon the free circuits available, they lack the intelligence to switch the transmission from the crowded part of the spectrum to a more open and less occupied region.

Many of our wireless networks lack decision-making capabilities. They are not aware of the environment and they lack frequency-spectrum sensing capabilities. They are not intelligent enough. This is where the concept of CR comes into picture.

In the 1999 paper that first coined the term cognitive radio, Joseph Mitola III defines a cognitive radio as [Mitola, 1999]: “A radio that employs model based reasoning to achieve a specific level of competence in radio-related domains.”

Coming from a background where regulations focus on the operation of transmitters, the FCC has defined a cognitive radio as [FCC, 2005]: “A radio that can change its transmitter parameters based on interaction with the environment in which it operates.”

Meanwhile, the other primary spectrum regulatory body in the US, the NTIA adopted the following definition of cognitive radio that focuses on some of the applications of cognitive radio [Comments of NTIA on FCC ET Docket No. 03–108, 2005]: “A radio or system that senses its operational electromagnetic environment and can dynamically and autonomously adjust its radio operating parameters to
modify system operations, such as maximize throughput, mitigate interference, facilitate interoperability, and access secondary markets.”

The international spectrum regulatory community, in the context of the ITU-R WP 5A working document, has worked towards a definition of cognitive radio that focuses on the following capabilities [ITU-R WP 5A, 2011]: “A radio or system that senses and is aware of its operational environment and can dynamically and autonomously adjust its radio operating parameters accordingly.”

While aiding the FCC in its efforts to define cognitive radio, IEEE USA offered the following definition [IEEE USA, 2003]: “A radio frequency transmitter/receiver that is designed to intelligently detect whether a particular segment of the radio spectrum is currently in use, and to jump into (and out of, as necessary) the temporarily-unused spectrum very rapidly, without interfering with the transmissions of other authorized users.”

The IEEE also tasked the IEEE 1900.1 group to define cognitive radio, which has the following working definition [IEEE 1900.1, 2008]: “A type of radio that can sense and autonomously reason about its environment and adapt accordingly. This radio could employ knowledge representation, automated reasoning and machine learning, mechanisms in establishing, conducting, or terminating communication or networking functions with other radios. Cognitive radios can be trained to dynamically and autonomously adjust its operating parameters.”

Likewise, the SDR Forum (now the Wireless Innovation Forum) participated in the FCC’s efforts to define cognitive radio and has established two groups focused on cognitive radio. The Cognitive Radio Working Group is focused on identifying enabling technologies and uses the following definition [SDR Forum, 2007]: “A radio that has, in some sense, (1) awareness of changes in its environment and (2) in response to these changes adapts its operating characteristics in some way to improve its performance or to minimize a loss in performance.”

However, the SDR Forum Special Interest Group for Cognitive Radio, which is developing cognitive radio applications, uses the following definition [SDR Forum, 2007]: “An adaptive, multi-dimensionality aware, autonomous radio (system) that learns from its experiences to reason, plan, and decide future actions to meet user needs.”

The Virginia Tech Cognitive Radio Working Group has adopted the follow-
ing capability-focused definition of cognitive radio [Virginia Tech Cognitive Radio Work Group, 2011]: “An adaptive radio that is capable of the following:

1. Awareness of its environment and its own capabilities,
2. Goal driven autonomous operation,
3. Understanding or learning how its actions impact its goal,
4. Recalling and correlating past actions, environments, and performance.”

Finally, in his recent popularly cited paper that surveyed the state of cognitive radio, Simon Haykin defines a cognitive radio as follows [Haykin, 2005], which is the definition we adopt in this work:

**Definition 1.2.** CR: An intelligent wireless communication system that is aware of its surrounding environment (i.e., outside world), and uses the methodology of understanding-by-building to learn from the environment and adapt its internal states to statistical variations in the incoming RF stimuli by making corresponding changes in certain operating parameters (e.g., transmit power, carrier frequency, and modulation strategy) in real time, with two primary objectives in mind:

1. **Highly reliable communications whenever and wherever needed;**
2. **Efficient utilization of the radio spectrum.**

All these definitions assume that cognition will be implemented as a control process, presumably as part of a software-defined radio. Second, all of the definitions at least imply some capability of autonomous operation. Finally, the following are some general capabilities found in all of the definitions:

- **Observation**: whether directly or indirectly, the radio is capable of acquiring information about its operating environment.
- **Adaptability**: the radio is capable of changing its waveform.
- **Intelligence**: the radio is capable of applying information towards a purposeful goal.
The CR concept enables a network or wireless node to change its transmission or reception parameters to communicate efficiently by avoiding interference with other users. With this technology, the CR actively monitors the external wireless network environment that affects the propagation of radio signals, user behavior, the network states, and internal factors such as system hardware, sensors, etc. Based on the results gathered, the cognitive engine asks the system to alter the transmission and reception parameters. CR will prevent traffic jams in wireless networks by routing transmissions of signals from occupied frequency bands to idle frequency bands.

CR will be an addition to the existing network infrastructure where both electronic devices and the network will share intelligence. Although CR is still in its initial stages of development and has limited implementation in consumer products, it is a very popular ongoing topic of academic and industrial research. Certain military and public safety services are already making use of CRs today. These are useful in emergency and extreme conditions when interoperability and guaranteed quality of service are essential. CRs also offer users autonomous operations.

Today, the baseband processing capabilities of handheld devices have advanced to a level where one processor can process the various waveforms that may be used across many frequency bands. Along with advancements in capabilities of baseband processors, there are certain essential technologies upon which progress in the field of CR depends. One of these key technologies is better cognitive algorithms that can manage the retrieval and storage of portable data.

Although there are technological barriers that must be overcome, the core enabling technologies are emerging, thus giving cause for optimism that the days of wireless networks based on CR technology are not very far off. Soon, our handheld devices will have the capability of sensing the network and changing roles and functionality depending on the network.

Based on the definitions we presented above, CRs are radios that can intelligently sense the radio environment, interpret it, plan the most appropriate system response (according to preset rules), and react accordingly. A more specific and application-related definition would be that CRs provide a method of wireless communication in which the wireless node modifies its reception and transmission parameters to communicate via efficient use of bandwidth while avoiding interfer-
ence with other users. Using this definition, the three attributes that are critical for functional CRs are frequency sensing, modulation classification, and implementation of higher level protocol analysis.

In summary, CRs control how the radio behaves, where SDRs determine how the radio is constructed and controlled.

1.3 Contributions of This Work

In this work, we design a modulation classification (MC) method that can receive a signal in the passband and, through multiple steps, automatically determine its type of modulation. In the large scheme of CRs and considering the definition for CR adopted in this work, our MC method falls in the area of first step of CR: observing and acquiring information about its operating environment. As seen in Figure 1.2, after observing the environment, we need to analyze this information. Part of the information concerning the radio environment is the type of modulation and other parameters that are required in order to make a decision on how to demodulate and acquire the data, and finally adapt the radio system to it.

Here we discuss methods and algorithms that would improve the current methods and introduce a general algorithm with expandability for future applications and modulation schemes.

For this purpose, we extensively study the available Gaussianity tests to find the most appropriate test to classify multi-carrier modulations from single-carrier modulations. We achieve this goal through various simulations, considering all possible elements and employing the Monte Carlo method. This provides us with a strong first step in the algorithm to classify the received signal. Our simulations provide us with a test that would not only perform best against the present noise, but also would give us the fastest processing time. In doing so, we also optimize the chosen test for this particular purpose.

We also design a special case of $k$-means algorithm, through applying greedy $k$-center algorithm for initialization, to classify various single-carrier modulations. By applying $k$-center algorithm before $k$-means, a significant performance improvement is achieved. By defining a new parameter, we are able to measure how much improvement it brings to combine the two clustering algorithms. Furthermore, we
use this parameter to show that this method gives high accuracy in classifying $M$-PSK, $M$-QAM, and various other modulation types.

We further demonstrate that utilizing this algorithm in OFDM modulations can provide us with a tool to correct any imbalances and offsets in an OFDM signal before demodulating it. Applying this method enables the receiver to determine any frequency and timing offsets and take the necessary actions to correct it. This would help us in synchronizing OFDM signals and may improve the throughput of the signal by reducing the necessary overhead of the transmitted signal.

To complete our algorithm we use additional tests, including autocorrelation and cyclostationarity tests, to blindly extract the parameters of an OFDM signal, and automatically determine the most suitable demodulation process necessary for the extracted modulation type.

Figure 1.3 depicts our proposed algorithm for modulation classification. Blocks in purple indicate the aforementioned steps and our contributions in this algorithm. The section number in each block refers to the section of this dissertation where we have discussed each of these steps.
Figure 1.3. Detailed steps in the proposed algorithm for modulation classification. The blocks in purple include the contributions of this research.

1.4 Outline of the Thesis

In Chapter 2, we introduce and formulate the problem for this thesis. We also take a look at the work that has been done in this area and we overview the methods in the literature.

Chapter 3 reviews some of the currently available tools for our algorithm and investigates some of the previously used methods. We also review the work we have done to choose among these methods.

In Chapter 4, we present our methodology. We mainly focus on designing the classification method using In-phase-Quadrature (I–Q) diagrams. We also address the problems that may occur in single-carrier and multi-carrier modulation
classification and propose solutions. We address some of the issues in this method, such as synchronization and carrier recovery, especially in the case of multi-carrier modulations and use the I–Q diagrams to overcome these issues.

Chapter 5 shows the performance of the model under different propagation environments and test conditions. Also in this chapter, hardware implementation of the algorithm, using Python and GNU Radio, on Universal Software Radio Peripheral (USRP) boards are discussed and some simulations using the hardware on real signals is presented.

Chapter 6 presents our conclusions and outlines some directions for future work on this project.

In Appendix A, the background of Gaussianity tests are explored more deeply and the skewness and kurtosis of distributions as a mean of Gaussianity test is discussed.

Appendix B details the derivation for the final version of formulas used in this thesis as the Giannakis–Tsatsanis test of Gaussianity. These contain some minor corrections to the test that is introduced in the literature.

Finally, in Appendix C, we study the algorithms in cluster finding more thoroughly and overview the available algorithms.
Chapter 2

Background

2.1 Introduction

In this chapter, we define the problem that we have considered in this work and overview previous work that is discussed in the literature. We categorize different available methods and give a brief overview of their performance, advantages and disadvantages.

2.2 Modulation Classification: Problem Formulation

Modulation classification (MC) is the process of recognizing the type of signal modulation in use with minimum or no a priori knowledge [Grimaldi et al., 2007]. Software-defined MC automatically detects the type of modulation and applies the necessary demodulation techniques to the signal in order to retrieve the message.

Definition 2.1. Given a measurement \( x(t), 0 \leq t \leq \tau \), a modulation classifier is a system that recognizes the modulation type of \( x(t) \) from \( M \) possible modulations, \( \{i_1, i_1, \cdots, i_M\} \).

The received signal \( r(t) \) is typically considered as a modulated signal received through a communication channel, and corrupted by additive noise, i.e.,

\[
r(t) = x(t) * h(t) + n(t) \tag{2.1}
\]
where $x(t)$ is the original transmitted signal, $\ast$ depicts convolution, and $h(t)$ is the impulse response for the whole signal path that includes transmit pulse-shaping, communication channel, and the receiver antenna before the demodulation process begins.

Most of the current modulation classifiers only cover a few modulation schemes and generally a few single-carrier signals. But today, the ever-increasing applications of multi-carrier modulations, such as OFDM (orthogonal frequency division multiplexing), require the design of classifiers that would automatically detect a vast number of modulations without having any \textit{a priori} information about them.

There are three main steps in every MC process:

1. Measurement,
2. Feature extraction, and
3. Decision.

The feature extraction and decision functions of the MC process are performed by modulation detection algorithms. Categories of modulation detection algorithms include:

1. Likelihood-based (LB), which include \cite{Panagiotou et al., 2000; Dobre et al., 2007}
   
   (a) Average likelihood ratio test (ALRT): unknown quantities are treated as random variables, and the likelihood function (LF) is computed by averaging over them. ALRT provides an optimal solution in the sense that it minimizes the probability of misclassification; however, it requires high computational complexity in many cases of practical interest.

   (b) Generalized likelihood ratio test (GLRT): unknown quantities are treated as deterministic unknowns, and the method employs maximum likelihood (ML) estimates of these quantities to compute the LF. GLRT has the disadvantage of failing to identify nested signal constellations, such as $M$-QAM

   (c) Hybrid likelihood ratio test (HLRT): this method is a combination of ALRT and GLRT, in which some of the quantities are treated as random
variables and some as deterministic unknowns. The nested signal constellation problem is overcome by averaging over the signal constellation points to calculate the LF;

2. Feature-based (FB), which includes

(a) instantaneous amplitude, phase, and frequency,
(b) wavelet transform, and
(c) signal statistics [Azarmanesh and Bilén, 2007].

In general, MC can be approached either from a decision-theoretic or a pattern recognition framework [Hsue and Soliman, 1990]. The decision-theoretic approach is based on composite hypothesis testing; the classifier resulting from this hypothesis is optimal and it minimizes a predefined cost function, e.g., the average error rate. The complexity of this approach is highly dependent on the number of unknown parameters associated with the received waveform. In the pattern recognition approach, the classifier discriminates various modulation types based on a chosen feature space. A feature extraction system is usually required to map the incoming signal into the feature space.

Many issues concerning the implementation of these methods more or less have been solved. For most of these methods, however, accurate preprocessing is required for effective implementation of MC algorithms. Thus, devising low complexity blind algorithms and MC methods that rely less on preprocessing is a next step in this research domain. Also, new MC issues are occurring as a result of new wireless technologies such as OFDM and Multiple Input–Multiple Output (MIMO). So, a desirable classification method would have the capability to

1. be easily applied and expanded to new modulation schemes, and
2. use unique parameters of modulation schemes for classification, those that would act as a fingerprint for each scheme.

Such a software modulation classifier would enable the automatic selection of the right demodulation for each modulation type as depicted in Figure 2.1.
Figure 2.1. Automatic modulation classification for choosing the correct demodulation method.

A modulation classifier should be considered as a subsystem of a closed-loop feed-back system, which includes a parameter estimator, equalizer, modulation classifier, and demodulator.

Figure 2.1 shows the simplest situation, in which the MC makes its best decision using the information provided to the classifier, without having to investigate the quality of the information. In this simple case, one assumes that the channel is equalized perfectly and all the signal parameters are known.

In the whole system scenario, each subsystem is itself a complex system. As such, the optimization of this entire system can be very challenging. Research has not covered this situation in the entirety, thus it is an interesting direction for future research.

Here we will only cover modulation classification and MC-oriented parameter estimation.

2.2.1 Assumptions Concerning SNR, Sampling Rate, and Time and Carrier Recovery

In this work, we extensively use SNR as one of the important parameters in determining the quality of the passband signal. SNR in this dissertation is the signal-to-noise ratio of the received signal after the received filters but before the A/D converter and is an indication of the quality of the communications channel:

\[
\text{SNR} = 10 \log_{10} \left( \frac{P_{\text{signal}}}{P_{\text{noise}}} \right). \tag{2.2}
\]
Also, as we explain in Chapter 4, we assume the signal to be sufficiently oversampled. Oversampling at rate $f_s \geq 4B$, where $B$ is the monolateral signal bandwidth (i.e., $[-B, B]$), eliminates aliasing in the cyclic frequency domain. Oversampling at this rate is essential in detecting multi-carrier signals in our algorithm.

We also assume an ideal A/D converter (i.e., no additional noise because of A/D, including dither), perfect timing and carrier recovery, and sampling rate of one per symbol at the output of the matched filter. Thus, whenever we refer to the number of samples being used in our simulations, especially our clustering simulations, we refer to samples taken from the output of the matched filter.

2.3 Background: Existing Methods

2.3.1 General MC Algorithms

The current state of the art in modulation classification is the decision-theoretic approach, which uses appropriate likelihood functions or approximations.

Dobre et al. [2005] gives a systematic review of a number of existing MC techniques that were developed prior to its publication. In Dobre et al. [2007], they complete their review, give a survey on existing modulation classification techniques, and present some classical approaches and new trends.

In this section, we also overview some of the methods that have been proposed for modulation classification and describe their development in recent years.

Initial attempts to implement MC algorithms occurred circa 1980. Aisbett [1987] used analog modulated-signal time-domain parameters to accomplish automatic modulation recognition. Whelchel et al. [1989], for the first time, used an artificial intelligence method (Neural Networks), as opposed to maximum likelihood method, to perform MC. They proposed a general demodulator and compared their results to the results from the maximum likelihood method.

Another method has been proposed for $M$-ary PSK signals in AWGN channels in Yang and Soliman [1997] using phase density functions of MPSK signals. This also performs very well but only in relatively high SNR channels.

Liedtke [1984] used both the decision-theoretic approach and the pattern recognition approach for digitally modulated signal discrimination, claiming that error-
free MC was achievable at SNR \( \geq 18 \text{ dB} \).

Moment-based classifiers, using various statistical features of the signal, have been extensively used as a means of modulation classification.

*Hsue and Soliman* [1990] identified a statistical moment-based classifier (SMBC). *Davidson et al.* [2004] present a new set of signal features that use the circular statistics of the signal to distinguish between FSK and QAM signals. *Guan et al.* [2004] introduce three spectogram-based MC algorithms. It uses principal component analysis (PCA) to achieve acceptable accuracy in SNR as low as 2 dB. *Shimbo et al.* [2007] linearly transform amplitude and phase of a received signal, and then use joint moments to classify modulations.

*Shen et al.* [2006] also use modulation classification based on fourth-order cumulants for \( M \)-PSK signals. *Liu and Xu* [2006] propose a similar algorithm but using different higher order cumulants: combinations of fourth- and eighth-order cumulants. *Mirarab and Sobhani* [2007] introduce a classifier based on higher order cumulants for classifying PSK, QAM, and ASK modulations. All of these algorithms achieve similar results with respect to accuracy but differ slightly in complexity.

*Park et al.* [2007] discuss an MC method capable of classifying both analog and digital modulations, using seven statistical signal features. This classifier studies classification on five analog and four digital modulations, including FSK and PSK, and achieves 95\% success rate at SNR ratios as low as 5 dB.

*Shi et al.* [2008] propose a hybrid MC scheme based on blind time synchronization, differential processing, and cumulants to make asynchronous classification for high-order QAMs. This problem assumes an unknown timing error, unknown frequency offset, and unknown frequency shift for the signal, but achieves acceptable results for these situations only for SNR larger than 20 dB.

One of the other popular methods in the literature has been the use of Maximum Likelihood (ML) for modulation types. *Wei and Mendel* [1995] have formulated a likelihood-based approach to MC that is not limited to any particular modulation class. Their approach is the closest to a constellation-based MC. However, carrier phase and clock recovery have not been addressed. *Lin and Kuo* [1997] have also proposed a sequential probability ratio test in the context of hypothesis testing to classify among several QAM signals. Their approach is novel in the sense
that new data continuously updates the evidence.

Abdi et al. [2004] use a quasi-HLRT classifier along with an antenna array to improve the accuracy of HLRT method over the single-antenna classifier scenario. Yucek and Arslan [2004] present a sub-optimum ML classifier for adaptive OFDM systems that has less complexity compared to a traditional ML classifier. In Leinonen and Juntti [2004], another classifier for the same task is introduced that uses a quasi-log-likelihood ratio (QLLR) test. In Li et al. [2005], a quasi-HLRT classifier with unknown carrier frequency offsets is considered. All of these methods try to achieve the good performance of ML classifiers while trying to reduce the complexity of such classifiers.

Dobre and Hameed [2006] compare classifiers based on the hybrid likelihood ratio test (HLRT) and quasi-HLRT and discuss their performance versus computational complexity in fading channels. They argue that quasi-HLRT is less complex but can fail to estimate the channel parameters in low SNR channels.

In Puengnim et al. [2008], a study on a Bayesian classifier is presented that recognizes Gaussian Minimum Shift Keying (GMSK) modulation with different bandwidths, according to the maximum a posteriori (MAP) rule. In Shi and Karasawa [2008], a combination of blind time synchronization, differential processing, and ML-based classifier is considered for unsynchronized QAM modulations. Mosquera et al. [2008] use ML for estimation of symbol rate of a linearly modulated signal, with applications in MC and signal monitoring. It studies this method in low SNR scenarios. Su et al. [2008] introduce a real-time MC based on ML, using a look-up table (LUT) to assign any unknown signal symbol to an address on the table. This performance achieves a high performance as it only needs a few additions to get the memory address index.

Umebayashi et al. [2006] propose a method for M-ary PSK in AWGN channels using minimum Hellinger distance (MHD). The MHD-based MC method is an alternative to the ML methods and, as stated by Umebayashi et al. [2006], in regards to implementation, gives slightly better results compared to ML methods.

Finally, for ML classifiers, Roufashbaf and Nelson [2009] first use deterministic particle filters to estimate the communication channel. They then performs Maximum Likelihood MC. Combining these, they achieve a modulation classification using as few as 50 data samples. The algorithm has been tested for BPSK, 4-QAM,
and 16-QAM and achieves acceptable results.

One of the other considered methods for designing classifiers is the use of
classification. Their database of modulations have been limited to 2- and 4-ASK/PSK/FSK modulations, but they have achieved classification performance of over 90% at SNR = 10 dB. *Wei and Mendel* [1999] use a fuzzy logic classifier in non-ideal environments, in which using precise probabilistic methods are difficult.


Wavelets and Support Vector Machine (SVM) have also been two of the popular
methods considered for MC in recent years. One of the first cases was *Ta* [1994]. They use the energy vectors derived from wavelet packet decomposition as feature vectors to distinguish between ASK, PSK, and FSK modulation types.

In *Hong and Ho* [2000], a wavelet transform for MC has been proposed. A
method for classifying different PSK, QAM, ASK, and FSK modulations using wavelet and wavelet support vector machine (WSVM) was introduced in *Dan et al.* [2005]. This method shows success rates greater than 95% even in low SNRs and is faster compared to similar methods in that family; however, it is very complex, but though much slower than common feature-based or likelihood-based methods, it is faster than the traditional SVM.

*Fucai and Yihua* [2007] introduce a feature extraction method for MC based on signal Wavelet Packet Transform Modulus Maxima Matrix (WPTMMM) and a novel SVM Fuzzy Network (SVMFN). This method has a high complexity and it is limited to nine digital modulations that are more common in satellite communications, but the results presented in this work show that it can achieve an accuracy of 98% in SNR as low as 0 dB.

*Meng and Si* [2007] design an MC algorithm and extract parameters by analyzing the relationship between wavelet transform amplitudes of the received signals. It includes two decision blocks and achieves 97% accuracy for SNR as low as 5 dB.

*Park et al.* [2008] present a MC method for digital signals without *a priori* information using wavelet key features and SVM. The results indicate a success
rate of 95% at SNR = 10 dB in SVM–DDAG classifier on an AWGN channel. Zhou et al. [2008] present a MC algorithm for M-PSK signals based on Kernel Fisher Discriminant Analysis (KFDA). It uses the fourth-order cumulants of the signal. From two classifiers based on the kernel function, it is shown in this work that KFDA achieves the same classification accuracy as SVM and requires less time.

In Gao et al. [2009], a combined modulation classification is presented for digital television communication systems. It uses multi-class SVM and fuzzy integral to build the classifier. The results show a good classification rate even in low SNR.

Apart from these methods, there has also been a few other methods that have used a hybrid of methods for MC or have suggested other ways that do not necessarily fall in any of the aforementioned categories. For example, da Silva et al. [2008] propose using distributed signal sensing to increase the probability of signal detection and correct modulation classification of primary users of the spectrum of a cognitive radio system at the expense of requiring messages to be exchanged among the radios of the system. Ye et al. [2007] use instantaneous amplitude and phase of the digitized intermediate frequency signal. The accuracy of the classification exceeds 95% at SNR higher than 10 dB. Tao et al. [2006] propose using multifactorial dimensions of a signal as distinctive features.

With respect to tree structured algorithms, one of the works that uses a similar approach to ours is presented in Grimaldi et al. [2007]. In this work, an algorithm for MC is proposed that utilizes a tree structure form of classifying different modulations. The proposed method requires several routines and calculations, because it has separated the classification of amplitude modulations and angle modulations in single-carrier classification. This algorithm performs modulation classification processes in five different nodes in the tree, but may come with the burden of increased processing time. The results show that for SNR = 20 dB their method gives almost perfect results. De Vito et al. [2008] introduce a few additional steps in the algorithm presented in Grimaldi et al. [2007], and De Vito and Rapuano [2009] present some experimental results from the previously designed algorithm in Grimaldi et al. [2007].

Dobre et al. [2007] has summarized feature-based and likelihood-based classifiers that we discussed above in Tables 2.1 and 2.2.
Table 2.1. Summary of feature-based classifiers. The modulations mentioned here have been used in the simulations in original papers [Dobre et al., 2007].

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Features</th>
<th>Modulations</th>
<th>Channel(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azzouz and Nandi [1996]</td>
<td>maximum PSD of normalized centered amplitude, phase and frequency</td>
<td>2ASK, 4ASK, BPSK, QPSK, 2FSK, 4FSK</td>
<td>AWGN</td>
</tr>
<tr>
<td>Hsue and Soliman [1990]</td>
<td>variance of zero-crossing interval sequence, phase difference</td>
<td>UW, BPSK, QPSK, 8PSK, BFSK, 4FSK, 8FSK</td>
<td>AWGN</td>
</tr>
<tr>
<td>Yang and Soliman [1997]</td>
<td>PDF of phase</td>
<td>UW, BPSK, QPSK, 8PSK</td>
<td>AWGN</td>
</tr>
<tr>
<td>Hong and Ho [2000]</td>
<td>variance of Haar WT magnitude</td>
<td>QPSK, 4FSK, 16QAM</td>
<td>AWGN</td>
</tr>
<tr>
<td>Swami and Sadler [2000]</td>
<td>normalized and fourth order cumulants</td>
<td>BPSK, 4ASK, QPSK, 16QAM, 64QAM</td>
<td>AWGN</td>
</tr>
<tr>
<td>Dobre et al. [2003]</td>
<td>eighth-order cyclic cumulants</td>
<td>BPSK, QPSK, 8PSK, 16QAM, 256QAM</td>
<td>AWGN</td>
</tr>
</tbody>
</table>

Table 2.2. Summary of likelihood-based classifiers. The modulations mentioned here have been used in the simulations in original papers [Dobre et al., 2007].

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Features</th>
<th>Modulations</th>
<th>Channel(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sills [1999]</td>
<td>ALRT</td>
<td>BPSK, QPSK, 16QAM, 32QAM, 64QAM</td>
<td>AWGN</td>
</tr>
<tr>
<td>Wei and Mendel [1995]</td>
<td>ALRT</td>
<td>16QAM, V29</td>
<td>AWGN</td>
</tr>
<tr>
<td>Huan and Polydoros [1995]</td>
<td>quasi-ALRT</td>
<td>UW, BPSK, QPSK, 8PSK, 16PSK</td>
<td>AWGN</td>
</tr>
<tr>
<td>Hong and Ho [2003]</td>
<td>ALRT</td>
<td>BPSK, QPSK</td>
<td>AWGN</td>
</tr>
<tr>
<td>Panagiotou et al. [2000]</td>
<td>GLRT, HLRT</td>
<td>16PSK, 16QAM</td>
<td>AWGN</td>
</tr>
<tr>
<td>Chugg et al. [1995]</td>
<td>HLRT</td>
<td>BPSK, QPSK, OQPSK</td>
<td>AWGN</td>
</tr>
<tr>
<td>Hong and Ho [2001]</td>
<td>HLRT</td>
<td>BPSK, QPSK</td>
<td>AWGN</td>
</tr>
<tr>
<td>Hong and Ho [2000]</td>
<td>HLRT</td>
<td>BPSK, QPSK</td>
<td>AWGN</td>
</tr>
<tr>
<td>Abdi et al. [2004]</td>
<td>ALRT, quasi-HLRT</td>
<td>16QAM, 32QAM, 64QAM</td>
<td>flat fading</td>
</tr>
</tbody>
</table>
2.3.2 Gaussianity Tests

Our study of previous works in the field of Modulation Classification, especially in the classification of multi-carrier signals, reveals that a thorough study of the Gaussianity test as a best fit for this purpose has not been performed.

Grimaldi et al. [2007] use the Giannakis–Tsatsanis test [Giannakis and Tsatsanis, 1990]. There is no justification for use of this specific test and there is no comparison to similar methods that have been used in other articles, such as Cramer–von Mises test used in Abdi et al. [2004], and Li et al. [2006].

Also, Lobato and Velasco [2004] employ a skewness–kurtosis test statistic for normality testing in correlated data. Wang and Ge [2005] use second-order and higher order cumulants to derive a method for identifying OFDM signals from single-carrier signals in Rayleigh channels.

The test used in Grimaldi et al. [2007] uses a simplified version of the test in Giannakis and Tsatsanis [1990] that was used in Akmouche [1999]. Akmouche [1999] adapts the Giannakis–Tsatsanis test, which is based on fourth-order cumulants to specific case of digital modulations, to reduce the algorithm complexity. As a worst case scenario, they take 256-QAM versus 32-OFDM and they achieve a probability of detection of $P_d = 0.99$. They also state that, although multipath, distortion, and all the other perturbations make the signal more and more Gaussian, the gap between single-carrier and multi-carrier modulations seems to be enough to use this test for practical applications.


A few methods have also been proposed in Al-Smadi [2005], but they did not address their application in this new field of single-/multi-carrier modulation classification, nor do they compare its results to other proposed methods. In Thadewald and Büning [2007], they conduct an investigation of the power of several Normality tests: Jarque–Bera, Kuiper, Shapiro–Wilk, Kolmogorov–Smirnov, and Cramer–von Mises. This study was performed purely considering their statistical
2.3.3 Image Processing and Pattern Recognition Techniques

In feature-based modulation classification, use of pattern recognition techniques is one of the methods that has not received much attention, but now, with faster processors, it has the potential to be an efficient classifier method.

One of these methods, which is very similar to the method we are developing is discussed in Mobasseri [1999, 2000] using constellation shape as a stable modulation signature. They use this information in a ML classifier for MC of single-carrier modulations in AWGN channels and in the presence of carrier recovery errors. The method used in classification is a fuzzy $c$-means to find clusters in the I–Q diagram. In this method, as with our method, there is a need to estimate the number of cluster centers before applying the algorithm to the diagram.

In Mobasseri [1999], an initial number of clusters, larger than expected, is specified. There are two drawbacks to this method: (1) it still only covers the case of AWGN channels and fails in multipath environments, and (2) its fuzzy $c$-means algorithm requires a learning process and is too complex to be used in a real-time system.

However, dealing with MC problems as a problem well-suited for pattern recognition algorithms goes before the work in Mobasseri [1999]. There have been other attempts to extract optimal features from signals. Histograms derived from functions like amplitude, instantaneous phase, frequency, or combinations of these have been used by Dominguez et al. [1991] and Huan and Polydoros [1995]. Jondral [1989] propose a modulation classifier utilizing the pattern recognition approach for recognition of both analog and digital modulation types. They use instantaneous amplitude, phase, and frequency histograms as key features for classification.

Similar methods have also recently been considered by Ahmadi [2010] and Shahmohammadi and Nikoofar [2002]. In Shahmohammadi and Nikoofar [2002], they use constellation shape as classification feature, but this work also only considers AWGN channels and it only addresses three QAM modulation schemes for classification. In these cases, it shows a 90% correct classification in SNRs around 10dB.
Ahmadi [2010] utilizes a method similar to Mobasser [1999] in using fuzzy \( c \)-means clustering for modulation classification of PSK and QAM modulations. But it also uses a Two Threshold Sequential Algorithmic Scheme (TTSAS) clustering algorithm to find the center of clusters in QAM modulations and then uses a matching technique with standard templates to classify these modulations. This work also indicates that the algorithm can be expanded to be used in PAM, MFSK and MASK modulations.

2.3.4 OFDM Feature Extraction

In Li et al. [2006], a detailed algorithm for extracting the parameters of a received signal with OFDM modulation is described and some numerical results are offered. But in this case as well, the problem of an efficient Normality test is not addressed. Also, this method gives very good results in AWGN channels, but for wireless digital modulations, studying the cases of Rayleigh and Rician channels for this case would be a necessity.

In Huang et al. [2007], a MC method is proposed for OFDM signals that uses 2\(^{\text{th}}\), 4\(^{\text{th}}\), and 8\(^{\text{th}}\)-order cumulants of the signal to classify the subcarriers of an OFDM signal, but their method only considers PSK signals.

Andrieu and Duvaut [1996] give a test for cyclostationarity that does not require any periodicities of the statistics to be known. Thus, this gives a useful tool in extracting the parameters of OFDM signals. Ramkumar et al. [2009] propose an MC based on cyclostationary feature detection and a predictor-based recursive blind equalizer, used in conjunction. Lundén et al. [2009] present a work on cyclostationary spectrum sensing for cognitive radios. In this work, a test for the presence of cyclostationarity in the signal is presented that we have also used in our work. Finally, Gardner et al. [2006] present a concise overview of the literature on cyclostationarity.

Shi et al. [2007] propose a blind OFDM system parameters estimation method for SDR that sequentially estimates the sampling frequency, the number of subcarriers, the cyclic prefix length, as well as timing and frequency offsets in OFDM systems. This method, similar to most of the methods designed for OFDM, is based on a combination of the cyclostationarity test and correlation.
2.3.5 Hardware Implementation

There is little material in the literature to consider the performance of different MC algorithms when implemented in hardware and with real off-the-air signals. The following two cases have come to our attention and we give a brief overview of them here.

O’Shea et al. [2007] implement a few classification algorithms in GNU Radio. They decompose these algorithms into logical blocks and implement them in GNU Radio signal processing blocks. Then, these blocks are demonstrated in a Linux environment, using the Universal Software Radio Peripheral (USRP) as a radio front-end. A second USRP on an unconnected host computer is used to generate the transmitted signals. This work is significant in showing a real-world signal input rather than ideal signals generated in MATLAB.

Park and Kim [2006] implement five different classifiers on reconfigurable SDR, by programming them in a DSP (TMS320C6203), and adding an additional feature: occupied bandwidth. These classifiers are capable of recognizing nine different analog and digital modulations. This work shows that the performance of these MC algorithms agrees with numerical simulation results.
Modulation of Single-Carrier Signals vs. Multi-Carrier Signals

3.1 Introduction

In this section, we show the significance of Modulation Classification (MC) and the tools necessary to build our algorithm.

The method that we propose here uses a tree structure (i.e., hierarchical) and a series of check points to identify the type of modulation. The proposed algorithm can be seen in Figure 3.1 and consists of two main steps.

In the first step we determine whether we have a single-carrier or a multi-carrier signal. In the case of a single-carrier modulation, we use I–Q (in-phase–quadrature) diagrams of the received signals as unique features for classification, and apply a pattern recognition technique in the classifier to extract those features.

For the multi-carrier modulation case, a number of steps such as Gaussianity test, autocorrelation test, and cyclostationarity test are performed to identify and extract the parameters of the OFDM signal.

Before describing the details of the algorithm, we describe the tools and go over the types of modulations that we have used to test the algorithm. In this study, we are considering $M$-PAM, $M$-PSK, $M$-QAM, and OFDM (with subcarriers modulated as PSK and QAM) modulations.
3.2 Comparison of Modulations for the Purpose of Classification

To understand how the I–Q diagram of a signal can help in identifying the modulation scheme, we start with an overview of some of the common modulations currently being used in radio systems. All of these signals are a subset of digital modulations that have the following signal expression:

\[ x(t) = \Re \left\{ \sum_{k} A \hat{X}_k p(t - kT_s) e^{j(2\pi f_c t + \theta_c)} \right\}, \]

with \( kT < t \leq (k + 1)T, \ k = 0, 1, \ldots \), \( (3.1) \)

where \( T_s \) is the symbol duration, \( A \) is the signal amplitude, \( p(t) \) is a pulse-shaped function (which represents the impulse response of the overall signal path, including transmitter, channel, and receiver), \( f_c \) is the carrier frequency, \( \theta_c \) is the carrier phase, and \( \hat{X}_k \) assumes a value from a set of \( M \) complex numbers \( \{X_1, X_2, \ldots, X_M\} \).

Some of the digital modulation schemes that are discussed here are presented mathematically in the sections below [Azarmanesh and Bilén, 2007].

3.2.1 M-PSK (M-ary Phase Shift Keying)

PSK is the most common form of modulation currently found in digital communications. Symbols are distinguished from one another by phase changes, while the
amplitude remains constant. It can be written mathematically as

\[ x_{\text{PSK}}(t) = A \Re \left\{ \sum_k C_k e^{j2\pi f_c t} p(t - kT_s) \right\}, \]

with \( C_k = e^{j2\pi i/M}; i = 0, 1, \ldots, M - 1 \), \hspace{1cm} (3.2)

where \( A \) depends on the power of received signal, \( C_k \) maps the transmitted symbols, \( T_s \) is the symbol duration, \( f_c \) is the carrier frequency, \( M \) is the modulation level, and \( p(t) \) is a finite energy signal with a \( T_s \) duration. The symbol error rate vs. bit energy (SNR per bit, \( E_b/N_0 \)) for \( M \)-PSK signals is given by

\[ P_{s\text{MPSK}} = \text{erfc} \left[ \sqrt{\frac{\log_2(M)E_b}{N_0}} \sin \left( \frac{\pi}{M} \right) \right]. \] \hspace{1cm} (3.3)

### 3.2.2 \( M \)-QAM (\( M \)-ary Quadrature Amplitude Modulation)

QAM is the most popular type of modulation in combination with OFDM \[\text{[Nee and Prasad, 2000]}\]. The QAM representation that we consider here is known as square QAM or rectangular QAM and is the most common constellation of QAM. It can be written mathematically as

\[ x_{\text{QAM}}(t) = A \Re \left\{ \sum_k C_k e^{j2\pi f_c t} p(t - kT_s) \right\}, \]

with \( C_k = a_k + jb_k; a_k, b_k = 2i - M - 1; \]
\[ i = 0, 1, \ldots, M - 1 \], \hspace{1cm} (3.4)

where \( A \) depends on the power of received signal. Table 3.1 shows the normalization factors to ensure that the average energy over all symbols is one and, also shows the loss in \( E_b/N_0 \) compared to BPSK, which can also be translated into savings in power for QAM modulations over PSK.
The symbol error rate vs. bit energy for M-QAM signals is given by

\[ P_{\text{M-QAM}} = 2\left(1 - \frac{1}{\sqrt{M}}\right) \text{erfc}\left(\sqrt{\frac{3}{2(M-1)} \frac{\log_2(M)E_b}{N_0}}\right) \]

\[ - \left(1 - \frac{2}{\sqrt{M}} + \frac{1}{M}\right) \text{erfc}^2\left(\sqrt{\frac{3}{2(M-1)} \frac{\log_2(M)E_b}{N_0}}\right) \]

(3.5)

**Table 3.1.** QAM normalization factors and normalized Euclidean distance differences.

<table>
<thead>
<tr>
<th>Modulation</th>
<th>Normalization Factor</th>
<th>Maximum ( E_b/N_0 ) loss relative to PSK in dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPSK</td>
<td>( 1/\sqrt{2} )</td>
<td>0</td>
</tr>
<tr>
<td>QPSK (4-QAM)</td>
<td>( 1/\sqrt{2} )</td>
<td>0</td>
</tr>
<tr>
<td>16-QAM</td>
<td>( 1/\sqrt{10} )</td>
<td>3.98</td>
</tr>
<tr>
<td>64-QAM</td>
<td>( 1/\sqrt{42} )</td>
<td>8.45</td>
</tr>
<tr>
<td>128-QAM</td>
<td>( 1/\sqrt{85} )</td>
<td>13</td>
</tr>
</tbody>
</table>

### 3.2.3 PAM (Pulse Amplitude Modulation)

M-ASK (M-ary Amplitude Shift Keying), which is still sometimes used for its simplicity, is a subclass of modulations known as PAM. We can write a PAM signal as

\[ x_{\text{PAM}}(t) = \sum_k C_k p(t - kT_s), \]

(3.6)

where amplitude \( C_k \) is determined by the message data that are random.

The symbol error rate vs. bit energy for 4-PAM signals is given by

\[ P_{\text{4PAM}} = \frac{3}{4} \text{erfc}\left(\sqrt{\frac{2E_b}{5N_0}}\right). \]

(3.7)
3.2.4 OFDM (Orthogonal Frequency Division Multiplexing)

OFDM is a popular scheme for wideband digital communication that is used as a digital multi-carrier modulation method. The main advantage of OFDM over single-carrier modulations is its ability to cope with severe channel conditions (e.g., attenuation of high frequencies in a long copper wire, narrowband interference, and frequency-selective fading due to multipath) without the need for complex equalization filters. In its general form, it is expressed as

\[
x_{\text{OFDM}}(t) = A \Re \left\{ \sum_{k} \sum_{n=0}^{N_p-1} C_{n,k} e^{j2\pi n f_c t} \right\},
\]

with \( C_{n,k} \in \mathbb{C}; \mathbb{E}\{C_{n,k}\} = 0 \), \( (3.8) \)

where \( A \) depends on the power of received signal, \( C_k \) and \( C_{n,k} \) map the transmitted symbols, \( T_s \) is the symbol period, \( f_c \) is the carrier frequency, \( N_p \) is the number of OFDM subcarriers, \( M \) is the modulation level, \( p(t) \) is a finite energy signal with a \( T_s \) duration, and \( \mathbb{C} \) is the set of the complex numbers.

Simulation results of different modulation schemes show that constellation shapes (i.e., the I–Q diagram) as shown in Figures 3.2 through 3.4 are a global and stable signature for different digital modulations, making them a reliable way to classify different modulation schemes.

However, there are some modulation schemes that do not yield unique I–Q diagrams. These signal modulation schemes require different MC methods. The
Figure 3.3. I–Q diagram of PSK modulations: (a) 4-PSK, (b) 16-PSK, (c) 64-PSK.

Figure 3.4. I–Q diagram of PAM modulations: (a) 4-PAM, (b) 16-PAM, (c) 64-PAM.

I–Q diagram of these modulations is similar to some of the modulation schemes that we are considering for classification in this algorithm, therefore they must be considered.

Continuous Phase Modulations (CPM) are an example of these modulations. We will briefly present an example of these algorithms.

3.2.5 Continuous Phase Modulations (CPM)

3.2.5.1 MSK (Minimum Shift Keying)

MSK is an example of CPM, and can be written as

\[
x_{\text{MSK}}(t) = \sum_k A_{I_k}(t) \cos \left( \frac{\pi t}{2T} \right) \cos 2\pi f_c t + A_{Q_k}(t) \sin \left( \frac{\pi t}{2T} \right) \sin 2\pi f_c t \quad (3.9)
\]

MSK modulation can also be viewed as OQPSK (Offset QPSK) with a half
sinusoid for each bit, or as CPFSK (Continuous Phase FSK) with a frequency separation of half the bit rate. Its I–Q diagram can be seen in Figure 3.5.

3.2.5.2 GMSK (Gaussian MSK)

GMSK is a continuous-phase FSK modulation scheme. It is widely used in the GSM standard and is similar to standard MSK; however, the digital data stream is first shaped with a Gaussian filter before being applied to a frequency modulator. Its I–Q diagram is similar to I–Q diagram of MSK.

3.2.6 $M$-FSK ($M$-ary Frequency Shift Keying)

$M$-FSK transmits digital data by shifting the output frequency between $M$ predetermined values. $M$-FSK is not particularly spectrally efficient, but offers some advantages, e.g., immunity to amplitude noise, bit rate higher than baud rate, and constant transmit power. The $M$ different frequencies on which the transmitted
Figure 3.6. I–Q diagram of FSK modulations: (a) 4-FSK, (b) 16-FSK, (c) 64-FSK.

message is quantized are given by

$$x_{FSK}(t) = A \Re \left\{ \sum_k C_k e^{2\pi(jf_c + C_k) \cdot p(t - kT_s)} \right\},$$

with $C_k = i/2T_s; i = 0, 1, \ldots, M - 1$, \hspace{1cm} (3.10)

where $A$ is the amplitude of the received signal. The I–Q diagram of FSK signal can be seen in Figure 3.6.

3.3 Gaussianity Tests as Means of Classification

In an OFDM modulation, all orthogonal subcarriers are transmitted simultaneously. In other words, the entire allocated channel is occupied with the aggregated sum of the narrow orthogonal sub-bands. Thus, since it is a combination of multiple carriers, the OFDM-modulated signal can be considered to be a composite of a great number of independent identically distributed (IID) random variables. Therefore, using the central limit theorem (CLT),\(^1\) we can claim that the amplitude distribution of the sampled signal can be approximated with a normal (Gaussian) distribution. However, this cannot be said of the case of a single-carrier modulated signal [Li et al., 2006]. Hence, multi-/single-carrier classification can be made with a simple normality (Gaussianity) test.

\(^1\)The central limit theorem (CLT) states that the sum of a large number of IID random variables will be approximately normally distributed (i.e., follow a Gaussian distribution, or bell-shaped curve) if the random variables have a finite variance [Papoulis and Pillai, 2002].
A normal distribution can be expressed as follows:

\[
f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.11}
\]

where \(\mu\) is the mean and \(\sigma\) is the standard deviation of the distribution. We use Gaussianity tests as hypothesis testing to determine whether a signal’s samples are normally distributed or not.

Hypothesis testing is a statistical decision-making technique. These techniques rely on using the information in a random sample from the population of interest. If the result is consistent with the hypothesis, then the hypothesis is true. Else, the hypothesis is false. It must be emphasized that the truth or falsity of a particular hypothesis can never be known with certainty.

Hypothesis testing can be two-sided alternative or one-sided hypothesis. In either case, two statements are claimed: \(H_0\) and \(H_1\). The value \(H_0\) is referred to as the null hypothesis, while \(H_1\) is referred to as the alternative hypothesis. The decision will be either reject or fail to reject the null hypothesis.

In testing for departures from the normal distribution, the null hypothesis \(H_0\) is that the random variable \(X\) under consideration is distributed as a normal variable. If either \(\mu\) or \(\sigma\) is not specified completely, then the null hypothesis under consideration is a complete hypothesis. Here we deal with the complete null hypothesis with both \(\mu\) and \(\sigma\) unknown.

One must be mindful that cumulants, which are needed in these tests, involve expectations and cannot be computed in an exact manner from real data. Hence, they must be approximated. Therefore, we replace their true values with their sample averages.

Other tools for determining normality of a distribution include Histogram plot and skewness and kurtosis tests.

Histogram is widely used as an approximation to the PDF (Probability Density Function). It describes the number of times the estimator produces a given range of values. Therefore, histograms can be used as aids to selecting a probabilistic model, i.e., Gaussian or non-Gaussian.

Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right
of the center point. The kurtosis is also an important property of the density function. It measures the degree of peakedness or flatness of a distribution.

The skewness of a distribution is the third standardized moment, denoted by $\gamma_1$ and defined as

$$\gamma_1(y) = \frac{E\{y^3\}}{\sigma_y^3}. \quad (3.12)$$

The kurtosis is the fourth standardized moment and is formally given by

$$\gamma_2(y) = \frac{E\{y^4\}}{[E\{y^2\}]^2} - 3 = \frac{E\{y^4\}}{\sigma_y^4} - 3, \quad (3.13)$$

where $y$ is the random variable, $[\cdot]^2$ signifies taking the square of each element of the vector $[\cdot]$, and $E\{\cdot\}$ is the expectation operator. The normal distribution has the property that $\gamma_1 = 0$ and $\gamma_2 = 0$. These two parameters are very important and can be used in determining whether a sample distribution is normal. Further discussions on these two parameters can be found in Appendix A.

Normality tests have been discussed in the literature [Grimaldi et al., 2007; Mobasserí, 2000; Proakis, 2001] and a few of them have been proposed for this task. Although there is a vast number of tests available, some of them, such as $\chi^2$ test or Epps test, are not well suited for digital modulation due to their high noise sensitivity. The tests that have been recommended are modified versions of the aforementioned tests, Giannakis–Tsatsanis test as an example.

Next, we will review some of the normality tests that are being studied in this project. The approach we are taking to find the best suitable test here is to try each test in simulations under different conditions of noise and different types of modulations. The tests that are being considered are $\chi^2$, Jarque–Bera, Giannakis–Tsatsanis, Kolmogorov–Smirnov, Anderson–Darling, D’Agostino–Pearson, Shapiro–Wilk, Cramer–von Mises, and Lilliefors.

### 3.3.1 $\chi^2$ Test

The $\chi^2$ test, also known as chi–square or chi–squared test is defined by

$$d_{\chi^2} = \frac{1}{E_i} \sum_{i=1}^{N} (O_i - E_i)^2, \quad (3.14)$$
where $O_i$ are the observed counts and $E_i$ are the expected counts. The expected counts are calculated by

$$E_i = N(F(Y_{ui}) - F(Y_{li})),$$  \hspace{1cm} (3.15)

where $F$ is the cumulative distribution function for the distribution being tested, $Y_{ui}$ is the upper limit for class $i$, $Y_{li}$ is the lower limit for class $i$, and $N$ is the number of samples. The $\chi^2$ statistic has an approximate $\chi^2$ distribution when the counts are sufficiently large.

For the normal distribution, the hypothesized distribution is decomposed into a multi-normal distribution of $M$ cells, counting the number of observations in each cell and contrasting these, via a $\chi^2$ statistic or a likelihood ratio statistic, with the expected number of observations for each cell. The latter expected values are computed assuming the data did arise from a normal distribution [D’Agostino and Stephens, 1986].

The $\chi^2$ test is an alternative to the Anderson–Darling and Kolmogorov–Smirnov tests and it can be applied to both discrete and continuous distributions.

We are using the $\chi^2$ test as a reference to compare the other tests studied in this project. The code for this test used in simulations is the MATLAB function `chi2gof`. In this test, the number of cells is set to 10 by default, although as a rule of thumb, $2N^{2/5}$ can be a good starting point and every cell should have at least five data points.

### 3.3.2 Jarque–Bera Test

This test is actually the $\chi^2$ test with two degrees of freedom [Jarque and Bera, 1987], i.e.,

$$d_{JB} = \frac{N}{6} \left[ \gamma_1^2 + \frac{(\gamma_2 - 3)^2}{4} \right],$$  \hspace{1cm} (3.16)

where $\gamma_1$ is a measure of skewness and $\gamma_2$ is a measure of kurtosis and $N$ is the number of samples. The values for $\gamma_1$ and $\gamma_2$ can be calculated using Equations (3.12) and (3.13), respectively.

For large sample sizes, the test statistic has a $\chi^2$ distribution with two degrees of freedom. We accept the null hypothesis of having a normal distribution with the
error $\epsilon$ if and only if $d_{JB} < \chi^2_{2, \epsilon}$. The $\chi^2$ distribution table at the 0.05 confidence interval returns the number 5.99.\(^2\) Therefore, if the Jarque–Bera test statistic is greater than 5.99, the null hypothesis of normality is rejected. This is correct since the J–B statistic is usually only greater than 5.99 if the skewness and kurtosis are relatively far from zero, and the J–B statistic tends to be closer to one or less than one when skewness and kurtosis are close to zero.

Jarque–Bera test often uses the $\chi^2$ distribution to estimate critical values for large samples, deferring to the Lilliefors test for small samples. The MATLAB jbtest function, used in this project for simulation, by contrast, uses a table of critical values computed using Monte Carlo simulation for sample sizes less than 2000 and significance levels between 0.001 and 0.50. Critical values for a test are computed by interpolating into the table, using the analytic $\chi^2$ approximation only when extrapolating for larger sample sizes.

### 3.3.3 Giannakis–Tsatsanis Test

Giannakis–Tsatsanis test \cite{Giannakis and Tsatsanis, 1990} has the advantage of being relatively simple computationally, insensitive to signal shifts, and not requiring knowledge of the noise spectrum for pre-whitening. This statistical test, which is based on 4\(^{th}\)-order cumulants, has been adapted for digital modulations \cite{Akmouche, 1999} enabling the reduction of algorithm complexity.

However, before simplifying the computations, it must first be noted that the tested process must be normally stationary, although this is not the case for modulated signals, which are cyclostationary. But, the results presented by Akmouche [1999] justify that the test results remain suitable in our case. Hence, the adopted computationally simplified Gaussianity test offered by Giannakis and Tsatsanis [1990] is presented below. Another advantage of this test over the other tests is that it does not require setting a significance level when running the test.

The covariance matrix is defined as $\Sigma_c(u, v) = \text{cov}\{c(u), c(v)\}$ and the elements of vector $c$ are defined as follows:

\(^2\)5.99 is the $\chi^2$ statistic with two degrees of freedom and significance level of 0.05.
Real
\[ c_{4r}(0, t, t) = -\frac{1}{N} \sum_{i=t}^{N-t-1} x_r^4(i) + x_r^2(i) x_r(i + t) x_r(i - t) \] (3.17)

Imaginary
\[ c_{4j}(0, t, t) = -\frac{1}{N} \sum_{i=t}^{N-t-1} x_j^4(i) + x_j^2(i) x_j(i + t) x_j(i - t) \] (3.18)

where \( N \) is the number of samples, and \( x_r \) and \( x_j \) are the real and imaginary parts of the signal samples, respectively.

The test consists of four steps [Akmouche, 1999]:

1. Compute the above cumulants to form the vectors \( c_r \) and \( c_j \).
2. Then compute covariance matrix:
   \[ \Sigma_{c_r} = \text{cov}\{ c_{4r}(0, \eta_u, \eta_u) , c_{4r}(0, \eta_v, \eta_v) \} \] , (3.19)
   and
   \[ \Sigma_{c_j} = \text{cov}\{ c_{4j}(0, \eta_u, \eta_u) , c_{4j}(0, \eta_v, \eta_v) \} \] . (3.20)
3. Now, form the threshold
   \[ d_{G,4} = \sup \left( c_r^T \Sigma_{c_r}^{-1} c_r , c_j^T \Sigma_{c_j}^{-1} c_j \right) \] . (3.21)
4. Decide whether we have a single-carrier or a multi-carrier modulation depending on whether \( d_{G,4} < \tau_G \) holds, where \( \tau_G \) is an opportune set threshold, in which case the Gaussianity test is passed and multi-carrier modulation is confirmed.

The details on the derivation of the above formulas and the simplification process can be found in Appendix B. In these derivations, we have developed two different implementations of this test. The first has been mentioned by Akmouche [1999], and Grimaldi et al. [2007]. The second uses fewer calculations for computing the test statistic. Results from our implementation of the test do not show any significant differences with the results obtained from the first method. A sample
comparison of these two implementations can be seen in Figure 3.7. This figure shows the statistic for a sample size of 512 and is averaged over 100 runs of the simulation.

![Figure 3.7](image)

**Figure 3.7.** Comparing $d_{G,4}$ from the first and second implementations of Giannakis–Tsatsanis test for a randomly generated sequence.

Figures 3.8, 3.9, 3.10, and 3.11 show graphs of $d_{G,4}$ vs. SNR, obtained from the first and second implemented methods. The simulations use a maximum of 1000 samples for signal and average the results over 100 runs of the code in MATLAB. We choose a maximum of 1000 samples in these simulations because we want to use as few samples as possible for classification. In the next chapter it will be shown that, for the next step in our algorithm, 512 samples give reasonable results. Thus, in these simulations, we focus our attention on the number of samples close to that number. Due to randomness of the samples, these simulations may have unexpected peaks, as it is evident in Figure 3.9 for 400 sample size. For this reason, we average our results in 100 runs to avoid this random element in the
simulations. Figures 3.8 and 3.9 use a uniformly distributed randomly generated signal with additive white Gaussian noise as input. Figures 3.10 and 3.11 use a QPSK signal passed through an AWGN channel as the input. Figure 3.8 uses the first implementation for the Giannakis–Tsatsanis test and it shows that after normalization, the test gives similar results for different sample sizes with respect to SNR. In Figure 3.9 this is evident for second implementation of the test as well. In this figure, for sample size of 400, we see a peak at 10 dB. This is due to randomness of the signal and with further averaging of the results, we see that this peak disappears and the curves become more similar. In the case of Figures 3.10 and 3.11, the similarities between different sample sizes are more evident. However, the test statistic rises more slowly compared to testing it on randomly generated sequence.

These graphs are used to obtain the threshold for this test. Based on these graphs, we can set \( \tau_G \approx 0.13 \). For sample size of 512, this threshold would be sufficient and gives satisfactory results for SNR > 5 dB.

For this test, we have used our own MATLAB code.

### 3.3.4 Kolmogorov–Smirnov Test

The Kolmogorov–Smirnov test, also known as K–S test, is the most well-known distance test. Distance tests are based on the fact that the probability integral transformation

\[
F(x) = \int_{-\infty}^{x} f(y)dy
\]

(3.22)

can be used to convert the data to uniform distribution if the hypothesized distribution, \( f(x) \), is correctly specified. The concept of distance is drawn from the procedure, whereby the selected test statistic evaluates the degree of discrepancy between the transformed sample and uniform distribution.

The K–S statistic is defined as

\[
d_{KS} = \sqrt{N} \max_{1 \leq r \leq N} \left( F(x_r) - \frac{r - 1}{N} , \frac{r}{N} - F(x_r) \right)
\]

\[
= \sqrt{N} \sup_{0 \leq x_r \leq x_N} |F(x_r) - x_r| ,
\]

(3.23)

where \( N \) is the number of samples,
Figure 3.8. $d_{G4}$ vs. SNR, from the first implementation of Giannakis–Tsatsanis test for a randomly generated sequence: (a) raw data, (b) normalized data.

$r$ is the rank of the observation ($r = 1, 2, \cdots, N$),

$x_1 \leq x_2 \leq \cdots \leq x_N$, and

$F(x_r)$ is the value of the transformed variate.

The hypothesis regarding the distributional form is rejected if the test statistic, $d_{KS}$, is greater than the critical value obtained from a table. There are several variations of these tables in the literature that use somewhat different scalings for the K–S test statistic and critical regions. These alternative formulations should be equivalent, but it is necessary to ensure that the test statistic is calculated in a
Figure 3.9. $d_{G,4}$ vs. SNR, from the second implementation of Giannakis–Tsatsanis test for a randomly generated sequence: (a) raw data, (b) normalized data.

way that is consistent with how the critical values were tabulated.

Shapiro [1990] states that this test is of little use in any area other than theoretical studies because of its dependency on knowing the actual parameters of the distribution and also its relative weakness compared to similar procedures. However, since this is a very well-known test, for the purpose of completion of this review, we have included it in our simulations. For this test, we use the MATLAB function, kstest, for one-sample K–S test.
Figure 3.10. $d_{G,4}$ vs. SNR, from the first implementation of Giannakis–Tsatsanis test for a QPSK signal passing through AWGN channel: (a) raw data, (b) normalized data.

### 3.3.5 Anderson–Darling Test

Of the well-known distance tests, the Anderson–Darling statistic generally has the highest power in testing for normality for a wide range of alternatives when the parameters are not known [Shapiro, 1990]. This test also consists of four steps:

1. The data is ordered so that

$$x_1 \leq x_2 \leq \cdots \leq x_N$$
and the standardized variate is computed from
\[
\zeta_i = \frac{x_i - \overline{x}}{\sigma}, \quad i = 1, 2, \ldots, N,  
\]  
(3.24)
where \( x_i \) is the \( i \)th ordered observation, \( \overline{x} \) is the mean and \( \sigma \) is the standard deviation.

2. Next, the values for \( \zeta_i \) are converted to standard normal cumulative probabilities
\[
Z_i = P\{Z < \zeta\}, \quad i = 1, 2, \ldots, N.  
\]  
(3.25)
3. We then compute

\[ A^2 = \frac{-1}{N} \left\{ \sum_{i=1}^{N} (2i - 1) \left[ \ln(Z_i) + \ln(1 - Z_{N+1-i}) \right] \right\} - N, \tag{3.26} \]

where \( \ln(Z_i) \) is the natural log of \( Z_i \).

4. The final step is to compute

\[ d_{AD} = A^{2*} = A^2 \left( 1 + \frac{0.75}{N} + \frac{2.25}{N^2} \right). \tag{3.27} \]

This test is an upper-tailed test, meaning that, if the computed value of \( d_{AD} \) exceeds a certain threshold, then hypothesis of normality is rejected.

We have written our own code for this test in MATLAB.

### 3.3.6 D’Agostino–Pearson Test

D’Agostino–Pearson’s \( \kappa^2 \) test is based on transformation of the sample kurtosis and skewness, designed by D’Agostino and Pearson, suggesting a method of combining kurtosis and skewness [D’Agostino and Stephens, 1986]. This test and the Shapiro–Wilk test are specifically designed to detect departures from normality, without requiring that the mean or variance of the hypothesized normal distribution be specified in advance. These tests tend to be more powerful than the Kolmogorov–Smirnov test. The suggested statistic is

\[ d_{DP} = X^2(\gamma_1) + X^2(\gamma_2), \tag{3.28} \]

where the functions \( X(\gamma_1) \) and \( X(\gamma_2) \) are transformations of \( \gamma_1 \) and \( \gamma_2 \), respectively, which yield approximate standard normal variables. However, since these functions are not independent, \( d_{DP} \) does not have a \( \chi^2 \)-distribution with 2 degrees of freedom.

Therefore, it is necessary in performing the test to use the contours given by Bowman and Shenton [1975]. The test is to see if the pair \( (\gamma_1, \gamma_2) \) falls within the contour. If not, the normality assumption is rejected.

We have developed our own code for this test in MATLAB.
3.3.7 Shapiro–Wilks Test

The Shapiro–Wilks test is thought to be one of the best omnibus tests of normality [D’Agostino and Stephens, 1986, p. 406]. The Shapiro–Wilks $d_{SW}$ statistic is given by

$$d_{SW} = \frac{\left(\sum a_i x(i)\right)^2}{\left(\sum (x(i) - \mu)\right)},$$

(3.29)

where $x(i)$ is the $i^{th}$ order statistic, i.e., the $i^{th}$ smallest number in the sample from the observed sample of size $N$ and $\mu$ is the sample mean.

For this test we have used a code from the Mathworks’ website’s central file exchange. In this code the desired significance level, $\text{ALPHA}$, is an optional scalar input (default = 0.05). This test is generally considered relatively powerful against a variety of alternatives. The Shapiro–Wilks test is better than the Shapiro–Francia test for Platykurtic sample. Conversely, the Shapiro–Francia test is better than the Shapiro–Wilks test for Leptokurtic samples. When the series $X$ is Leptokurtic, this code performs the Shapiro–Francia test. Otherwise (when series $X$ is Platykurtic), the code performs the Shapiro–Wilks test. The observation number must exceed 3 and be less than 5000.

3.3.8 Cramer–von Mises Test

The Cramer–von Mises (C–V) statistic is defined by

$$d_{CV} = N \int_{-\infty}^{\infty} \left[F_N(\omega) - F(\omega)\right]^2 dF(\omega),$$

(3.30)

where $F(\omega)$ is the cumulative distribution function (CDF) of our distribution, $N$ is the size of random samples, and $F_N(\omega)$ is the empirical distribution function defined by

$$F_N(\omega) = \frac{\text{number of observations } \leq \omega}{N}, \quad -\infty < \omega < \infty.$$  

(3.31)

$^{3}$A distribution with negative kurtosis is called Platykurtic, or Platykurtotic and with positive kurtosis is called Leptokurtic, or Leptokurtotic.
Table 3.2. Percentage points for Cramer–von Mises test of Gaussianity with unknown $\mu$ and $\sigma^2$.

<table>
<thead>
<tr>
<th>Significance level $\alpha$</th>
<th>0.50</th>
<th>0.25</th>
<th>0.15</th>
<th>0.10</th>
<th>0.05</th>
<th>0.025</th>
<th>0.01</th>
<th>0.005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper tail</td>
<td>0.051</td>
<td>0.074</td>
<td>0.091</td>
<td>0.104</td>
<td>0.126</td>
<td>0.148</td>
<td>0.179</td>
<td>0.201</td>
</tr>
<tr>
<td>Lower tail</td>
<td>0.051</td>
<td>0.036</td>
<td>0.029</td>
<td>0.026</td>
<td>0.022</td>
<td>0.019</td>
<td>0.017</td>
<td></td>
</tr>
</tbody>
</table>

This is basically the integrated square error between the estimated cumulative distribution function and the measured empirical distribution function of the sample.

Using Probability Integral Transformation (PIT), the following formula to calculate C–V test statistic is obtained

$$d_{CV} = \sum_k \left\{ \Delta(k) - \frac{2k-1}{2N} \right\}^2 + \frac{1}{12N},$$

(3.32)

where $\Delta(k) = \Phi(\zeta)$, and $\Phi(\zeta)$ indicates the cumulative probability of a standard normal distribution and $\zeta$ is the standardized variate.

The test consists of the following steps [Li et al., 2006]:

1. Sample the incoming signal and take the real or imaginary part.
2. Arrange the samples in ascending order.
3. Estimate the sample mean and standard deviation.
4. Apply PIT by calculating the standardized variate $\zeta$.
5. Calculate the C–V test statistic using Equation (3.32).
6. Use the percentage points given by D’Agostino and Stephens [1986] to accept or reject normality hypothesis. These points are presented in Table 3.2.

Figure 3.12 shows the C–V statistic with respect to SNR for a single-carrier signal when passed through an AWGN channel. The result is obtained for a QAM signal with 512 samples and it is averaged over 100 runs of the simulation.
3.3.9 Lilliefors Test

This test is an adaptation of the K–S test. An interesting peculiarity of the Lilliefors test is the technique used to derive the sampling distribution of the criterion. In general, mathematical statisticians derive the sampling distribution of the criterion using analytical techniques. However, in this case, this approach fails and, consequently, Lilliefors decided to calculate an approximation of the sampling distribution by using the Monte–Carlo technique. Essentially, the procedure consists of extracting a large number of samples from a normal population and computing the value of the criterion for each of these samples. The empirical distribution of the values of the criterion gives an approximation of the sampling distribution of the criterion under the null hypothesis.

The statistic for this test is similar to that of Kolmogorov–Smirnov test and is given by

$$d_L = \max_i \left( F(x_r) - \frac{r-1}{N} , \frac{r}{N} - F(x_r) \right) .$$  \hspace{1cm} (3.33)

The test proceeds as follows [Lilliefors, 1967]:

\textbf{Figure 3.12.} Cramer–von Mises statistic vs. SNR for a QAM signal.
1. Estimate the population mean and population variance based on the data.

2. Find the maximum discrepancy between the empirical distribution function and the CDF of the normal distribution with the estimated mean and estimated variance. Just as with the Kolmogorov–Smirnov test, this will be the test statistic.

3. Finally, confront the question of whether the maximum discrepancy is large enough to be statistically significant, thus requiring rejection of the null hypothesis.

This is where this test becomes more complicated than the Kolmogorov–Smirnov test. Since the hypothesized CDF has been moved closer to the data by estimation based on those data, the maximum discrepancy has been made smaller than it would have been if the null hypothesis had singled out just one normal distribution. Thus the “null distribution” of the test statistic, i.e., its probability distribution assuming the null hypothesis is true, is stochastically smaller than the Kolmogorov–Smirnov distribution. This is the Lilliefors distribution. To date, tables for this distribution have been computed only by Monte Carlo methods.

For this test, we use the MATLAB function lillietest.

There are, of course, other tests in the literature as well, such as Kuiper V [Kuiper, 1960], Pyke’s C [Pyke, 1959], Brunk’s B [Brunk, 1962], Durbin’s D [Durbin, 1961], Durbin’s M² [Durbin and Knott, 1972], Watson’s U² [Watson, 1961], Fisher’s π [Fisher, 1928], Hartley–Pfaffenberger S² [Hartley and Pfaffenberger, 1972], and Geary’s Z [Geary, 1947]. However, these tests are special formats of aforementioned tests and are less known compared to the ones that are reviewed above.

3.4 Single-/Multi-Carrier MC Simulations

As seen in previous section, there is a number of ways for testing the Gaussianity to discriminate between single-/multi-carrier signals. However most of these methods were designed in statistics and were implemented prior to the full development and application of multi-carrier modulations. For example, in Pearson et al. [1977], a review of different Gaussianity tests has been presented and their powers have
been compared, but some of these tests behave very differently when processing modulated signals in presence of noise. Also, in the recent papers where these methods have been used, comparative tests have not been performed nor have there been efforts to modify these methods for this specific application.

In addition, different combinations of mean, standard deviation, skewness, and kurtosis can be implemented to study the possibility of designing a more efficient method for classifying single-/multi-carrier modulations. It must be noted that some of these tests are special forms of other tests; for example, Jaque–Bera and D’Agostino–Pearson are modified versions of \( \chi^2 \) test, and Lilliefors and Anderson–Darling are different versions of Kolmogorov–Smirnov test. However, each of them have been designed for a unique application.

The following figures present a few of our results. The simulations were produced using MATLAB R2009a x64 on Microsoft Windows7, using a system equipped with Intel Core i7 CPU and 12 GB of RAM.

Figures 3.14 through 3.16 show the error rates for the Gaussianity tests that we have obtained from simulations. These figures were created with a signal of length 512 and 50,000 runs of the simulation. For these simulations we have created three sets of signal scenarios:

- Signal consists of uniformly distributed random signal and normally distributed random noise, using MATLAB functions \texttt{rand} and \texttt{randn} (Figure 3.14).

- Signal consists of uniformly distributed random signal, after passing through AWGN, using MATLAB functions \texttt{rand} and \texttt{awgn} (Figure 3.15).

- Signal consists of modulated signal, after passing through AWGN, using MATLAB functions \texttt{modem}, \texttt{modulate}, and \texttt{awgn} (Figure 3.16).

The test in Figure 3.13 is performed to make sure that the threshold level for all these tests has been set properly, since they behave almost the same in this case. The error rate of the various tests on Gaussian signals are almost identical and constant with SNR, based on the first scenario. This is expected, because a Gaussian signal is not affected by Gaussian noise. This can also be used to verify that the tests are behaving as expected.
In Figure 3.14 we see that G–T test performs better in SNR $< 7$ dB but its performance degrades considerably in higher SNR in comparison to other methods. In low SNR, the performance of all these tests are lower than 70%. Among other methods, Shapiro–Wilk and Cramer–von Mises tests show superior performance. Figure 3.15 shows the performance of the tests in second scenario. The performances can be interpreted similarly to previous figure. G–T test shows better performance in SNR $< 14$ dB but success rate is lower than 80%. In this scenario as well, Shapiro–Wilk and Cramer–von Mises have better error rates compared to other tests. Finally, in Figure 3.16, the Shapiro–Wilk test has the best error rate. The G–T test also gives very low error rate but the results for it are not consistent with the results from other tests. This may be due to the threshold setting in this test.

Figures 3.17 and 3.18 show the timing results for these tests. To obtain the processing time, we used the MATLAB functions `tic` and `toc` at the beginning and end of our code. In these tests, the calculated times from left to right represent the following Gaussianity tests: (1) Jarque–Bera, (2) Kolmogorov–Smirnov, (3) Lilliefors, (4) $\chi^2$, (5) Shapiro–Wilk, (6) Anderson–Darling, (7) D’Agostino–Pearson, (8) Cramer–von Mises, and (9) Giannakis–Tsatsanis. It is seen that Jarque–Bera test has the worst processing time in all scenarios. The Cramer–von Mises and Anderson–Darling tests have the fastest processing time over all. They consistently give better results compared to other tests. We see that the processing time for the Cramer–von Mises test is almost half as that of the Shapiro–Wilk test.

### 3.5 Summary

In this chapter, we constructed the foundation for the tools that we need to complete our MC algorithm. The main parts of this chapter are the overview of various existing common modulation schemes that we will need in the second step of this algorithm. In the second part, we overviewed various Gaussianity tests. Choosing the best suitable test will be essential for classifying single-/multi-carrier signals in the first step of the algorithm.

Based on our study of Gaussianity tests for the purpose of classifying single-carrier and multi-carrier signals, we see that our implementation of Giannakis–
Tsatsanis test gives a better performance in error rate versus SNR when SNR is low and especially for the case of real signals. However, for the case of low SNR, when SNR < 5 dB the results are not reliable and increased processing time is required.

Conversely, results for the second signal scenario that we have considered shows that, for SNR < 13 dB the Giannakis–Tsatsanis test can still be favorable over other tests. Whereas, for SNR > 13 dB other tests have better error rates.

Among other tests, Shapiro–Wilk and Cramer–von Mises tests show superior error rates. For the Kolmogorov–Smirnov test, we also observed good performance.
Figure 3.14. Error rate vs. SNR for Gaussianity tests, when dealing with a uniformly distributed random signal that has passed through AWGN channel.

that is in contrast with what was suggested in the literature, which indicated poor performance for the Kolmogorov–Smirnov test in the presence of noise. But, the obtained result could be due to a specific implementation of this test in MATLAB.

For processing time, the Cramer–von Mises test consistently gives better times compared to the others. Considering both processing time and error rate, the Cramer–von Mises test and the Shapiro–Wilk test outperform the rest and seems to be the best option for this purpose.
Figure 3.15. Error rate vs. SNR for Gaussianity tests, for the second scenario, when a randomly generated signal passes through AWGN channel.
Figure 3.16. Error rate vs. SNR for Gaussianity tests, for the third scenario, when a QPSK signal passes through AWGN channel.
New Method for Modulation Classification — Overview of the Method

4.1 Single-Carrier Modulation Classification

In the previous chapter, we discussed a classification method that discriminates single-carrier signals from multi-carrier signals. Various Gaussianity tests were studied and their performance on the received signals was demonstrated.

A failed Gaussianity test indicates a single-carrier signal, which will branch the process to that of classifying various types of modulations in a single-carrier signal. Also, after extracting parameters of the OFDM signal, we will need to identify and demodulate each subcarrier present in that signal. We are developing a procedure for classifying the single-carrier modulations using its constellation shape and some pattern recognition techniques. Figure 1.3 depicts the steps required to complete the modulation classification.

First, we are developing an algorithm that estimates the number of symbols in the signal and then, by using methods of $k$-means and $k$-center clustering, we find the overall shape of the modulation via the I–Q diagram, making it possible to automate the complete modulation classification process.
4.1.1 Clustering Algorithms

This procedure involves finding the center for clusters of data on the constellation shape, which yields the number of symbols in the modulation and the position of these symbols on the I–Q diagram.

Next, an analysis of the number of symbols and their locations on an I–Q diagram is performed to determine the modulation type. One method would be to compare them with a database of modulations that will determine the modulation type received. In this case, the error between the center of clusters and the corresponding symbols in the constellation is calculated and compared to a threshold for classification.

A second more automated and efficient method would be to calculate the distance between the symbols and determine the type of modulation by further using pattern recognition techniques. We address this method in Chapter 6 as future work and present some initial results.

As discussed above, for single-carrier modulations, a combination of two clustering algorithms is being used. The clustering algorithm that we are using here implements vector quantization. A vector quantization can be formalized in terms of an assignment rule \( x \rightarrow C(x) \) and a reconstruction rule \( C \rightarrow \mu^{(C)} \). So, the aim of this algorithm is to choose \( C(x) \) and \( \mu^{(C)} \) in order to minimize the expected distortion or cost [MacKay, 2003, pp. 285–292]:

\[
\text{cost} = \sum_x P(x) \frac{1}{2} [\mu^{(C(x))} - x]^2 ,
\]   \hspace{1cm} (4.1)

where \( P(x) \) is the probability distribution of \( x \).

With many clustering algorithms present for pattern recognition, a major challenge is to find the number of cluster centers for initializing the clustering algorithm. As mentioned before, choosing the number of the centers is imperative for the correct classification of modulations. The most common approach is to find the solution that minimizes the Schwarz Criterion, defined as

\[
\text{cost} + \lambda mk \log N ,
\]   \hspace{1cm} (4.2)

where \( m \) is the number of dimensions, \( k \) is the number of clusters, and \( N \) is the
number of records (data samples). Since we are operating on a plane, we can assume \( m = 2 \).

There are various clustering algorithms that are used in image recognition. A discussion on different types of clustering algorithms can be found in Appendix C.

Next, we provide a detailed explanation of the clustering technique that we use. This technique efficiently detects the center of clusters in each I–Q diagram. The main algorithms that are being used are greedy \( k \)-center clustering and \( k \)-means clustering, which, in combination, form our clustering algorithm.

In order to explain the \( k \)-center and the \( k \)-means algorithms, we need to first introduce a few definitions.

**Definition 4.1.** \( D_k \), the cluster size for cluster \( C_k \), is defined as the least value for which all points in \( C_k \) are

- within distance \( D_k \) of each other, or
- within distance \( D_k/2 \) of some point called the cluster center.

**Definition 4.2.** Partition \( S \) of set of points \( X = \{x_i\} \) is defined as having the following two conditions

- \( \bigcup S = X \)
- \( C_p \cap C_q = \emptyset \) if \( C_p \in S, C_q \in S, C_p \neq C_q \).

**Definition 4.3.** Cluster size of partition \( S \) is defined as

\[
D = \max_{k=1,\ldots,K} D_k.
\]

4.1.1.1 \( k \)-means

The \( k \)-means algorithm is our main algorithm of clustering. It is the most common and well known solution for clustering. In this algorithm we try to minimize the typical (average) distortion that we call \( \text{cost} \):

\[
\text{cost} = \min_C \left[ \sum_{k=1}^{K} \sum_{i:x_i \in C_k} (x_i - \mu_k)^T(x_i - \mu_k) \right].
\]
\[ \min_{C} \left[ \sum_{k=1}^{K} \sum_{i: x_i \in C_k} \|x_i - \mu_k\|^2 \right], \quad (4.4) \]

where \(\mu_k\) is the centroid for cluster \(C_k\), defined as
\[ \mu_k = \frac{1}{N_k} \sum_{i: x_i \in C_k} x_i. \quad (4.5) \]

In discussing the \(k\)-center algorithm, we will see that, unlike \(k\)-means, the \(k\)-center algorithm does not have a closed-form expression for the optimal center of cluster. For a single cluster \(C\), the distortion will be
\[ \text{cost}(C, \mu) = \sum_{\mu \in C} \|x - \mu\|^2. \quad (4.6) \]

This cost is minimized when \(\mu = \text{mean}(C)\), which is the same as Equation (4.5). If \(\mu \neq \text{mean}(C)\), then we have the following lemma [Arthur and Vassilvitskii, 2006a]:

**Lemma 4.1.** For any \(C \subset \mathbb{R}^d\), and any \(\mu \in \mathbb{R}^d\)
\[ \text{cost}(C, \mu) = \text{cost}(C, \text{mean}(C)) + |C| \|\mu - \text{mean}(C)\|^2. \quad (4.7) \]

**Proof.** Using
\[ \mathbb{E}\{X - \mu\}^2 = \mathbb{E}\{\|X - \mathbb{E}\{X\}\|^2\} + \|\mu - \mathbb{E}\{X\}\|^2, \quad (4.8) \]
we expand the right hand side:

\[
\text{R.H.S.} = \mathbb{E}\{\|X\|^2 + \|\mathbb{E}\{X\}\|^2 - 2X\mathbb{E}\{X\}\} + \{\|\mu\|^2 + \|\mathbb{E}\{X\}\|^2 - 2\mu\mathbb{E}\{X\}\} \\
= \mathbb{E}\{X\}^2 + \|\mathbb{E}\{X\}\|^2 - 2\mathbb{E}\{X\}\mathbb{E}\{X\} + \|\mu\|^2 + \|\mathbb{E}\{X\}\|^2 - 2\mu\mathbb{E}\{X\} \\
= \mathbb{E}\{X\}^2 + \|\mu\|^2 - 2\mu\mathbb{E}\{X\} \\
= \mathbb{E}\{\|X - \mu\|^2\}.
\]
Let \( X \) denote a uniform random draw from cluster \( C \). Then
\[
\mathbb{E}\{\|X - \mu\|^2\} = \sum_{x \in C} \frac{1}{|C|} \|x - \mu\|^2 = \frac{1}{|C|} \text{cost}(\mu)
\] (4.9)
and
\[
\mathbb{E}\{\|X - \mathbb{E}\{X\}\|^2\} = \frac{1}{|C|} \text{cost}(\text{mean}(C)).
\] (4.10)
By replacing Equations (4.8) and (4.9) in Equation (4.10), the equality in Equation (4.7) is obtained.

\[ \Box \]

**Lemma 4.2.** During the course of using the \( k \)-means algorithm, the cost monotonically decreases.

**Proof.** Let \( \mu_1^{(t)}, \ldots, \mu_k^{(t)}, C_1^{(t)}, \ldots, C_k^{(t)} \) denote the centers and clusters at the start of the \( t \)th iteration of \( k \)-means. The first step of the iteration assigns each data point to its closest center; therefore,
\[
D(C_1^{(t+1)}, \ldots, C_k^{(t+1)}; \mu_1^{(t)}, \ldots, \mu_k^{(t)}) \leq D(C_1^{(t)}, \ldots, C_k^{(t)}; \mu_1^{(t)}, \ldots, \mu_k^{(t)}).
\]
On the second step, each cluster is re-centered at its mean. By Lemma 4.1,
\[
D(C_1^{(t+1)}, \ldots, C_k^{(t+1)}; \mu_1^{(t+1)}, \ldots, \mu_k^{(t+1)}) \leq D(C_1^{(t+1)}, \ldots, C_k^{(t+1)}; \mu_1^{(t)}, \ldots, \mu_k^{(t)}).
\]

Based on the above conditions, the \( k \)-means algorithm alternates between these two steps:

- For a fixed set of centroids, optimize cost by assigning each point to its closest centroid, using Euclidean distance.
- Update the centroids by computing the average of all the samples assigned to it.
4.1.1.2 $k$-center

$k$-center clustering is similar to $k$-means, but uses a different optimization criterion. We use this algorithm to give the initial points for the $k$-means algorithm. $k$-center focuses on worst case scenario, especially when we have outliers. $k$-center clustering aims at minimizing $D$ and it is expressed as follows,

$$\text{cost} = \min_S D(S) = \min_S \left[ \max_{k=1,\cdots,K} \max_{i:x_i \in C_k} \|x_i - \mu_k\|^2 \right]. \quad (4.11)$$

It minimizes the worst case distance to centroid $\mu_k$. $\mu_k$ is called the centroid, but unlike $k$-means, it may not be the mean vector. In $k$-center clustering, among the clusters only the worst cluster matters, whose farthest data point yields the maximum distance to the centroid compared to the farthest data points of the other clusters.

Another formulation for the $k$-center clustering is shown below,

$$\text{cost} = \min_S \left[ \max_{k=1,\cdots,K} \max_{i,j:x_i, x_j \in C_k} L(x_i, x_j) \right], \quad (4.12)$$

where $L(x_i, x_j)$ denotes any distance between a pair of objects in the same cluster.

This $k$-center clustering runs in $O(NK)$ time, where $N$ is the number of samples (data points) and $K$ is the number of clusters. Using more advanced data structures, in Feder and Greene [1988], an algorithm is described to compute exactly the same clustering in only $O(N \log K)$ time. However, a greedy version of $k$-center algorithm, first introduced in Gonzalez [1985], is guaranteed to produce a clustering, in which the cost is at most twice the optimal value. The algorithm chooses points one at a time, starting with an arbitrary input as the first center. In each iteration, it chooses the farthest point from any previously chosen center to be the next centroid.

It can be proven that the greedy algorithm achieves an approximation with a

---

1 $O()$, called “big O notation”, is a notation used to characterize functions according to their growth rates. A description of a function in terms of this notation usually only provides an upper bound on the growth rate of the function.

2 $\log_a n$ and $\log_b n$ differ only by a constant multiplier, which in big O notation is discarded; thus $O(\log n)$ is the standard notation for logarithmic time algorithms regardless of the base of the logarithm.
factor of two [Li, 2007]. That is, if

\[
D^* = \min_{\mathcal{S}} \left[ \max_{k=1,\ldots,K} \left[ \max_{i,j,x_i,x_j \in C_k} L(x_i, x_j) \right] \right],
\]

then the greedy algorithm guarantees that \( D \leq 2D^* \).

We use the greedy \( k \)-center algorithm to get the approximate locations of the
cluster centers. Next, we use the results to initialize \( k \)-means and then use it to
improve the results to find the exact locations of the centers. This extra step to
use the greedy \( k \)-center algorithm reduces the error considerably, especially for
high SNR channels. Examples of the difference is shown in Figures 4.1 and 4.2.
In these figures, (a) and (b) are the performance of the algorithm when we use
\( k \)-center initialization; (c) and (d) are when we simply use \( k \)-means algorithm and
choose the initial points randomly.

In Figures 4.1 and 4.2 we have used the same set of samples for the simulations
and it is clear that \( k \)-center and \( k \)-means algorithms perform poorly when used
alone. However, in order to measure and quantize this improvement in perfor-
mance, we need to define an attribute that would help us in comparing different
methods.

For measuring the performance of the clustering algorithms and also to come
up with a parameter that would allow comparison and classification of modulation
schemes, we define a term called relative error or Cumulative Deviation Error
(CDE) and denote it \( \varepsilon_{\text{CDE}} \). This error shows the deviation of cluster centers that
are found by the algorithm from the actual modulation states, with respect to
number of samples used. We define this error as follows:

\[
\varepsilon_{\text{CDE}} = \sqrt{\frac{\sum_i (\mu_i - X_i)^2}{\sum_i X_i^2}},
\]

where \( \mu_i \) is the center of cluster and \( X_i \) is the corresponding symbol of \( i^{th} \) symbol in
a constellation with \( K \) symbols. As another alternative parameter, we also consider
\( \varepsilon_{\text{CDE}}^2 \) for classification, which yields slightly better performance. The results where
we compare the performance of these two parameters are presented in Chapter 5.
Defining CDE is part of our contribution in this dissertation.

Further investigation of the performance of these algorithms using CDE can
Figure 4.1. Difference in the performance of $k$-means algorithm for 4-QAM and 16-QAM modulations, [(a), (b)] when initialized with $k$-center algorithm, and [(c), (d)] when initialized with uniformly distributed random samples. The simulated signal has 256 samples with SNR = 7 dB.

be seen in Figures 4.3, 4.4, and 4.5. The simulations have been performed for up to 10,000 sample sizes and averaged over 100 runs. The signal has a 16-QAM modulation with SNR = 5, 15, and 30 dB, respectively. These figures compare the relative error and clearly show that the $k$-center algorithm yields remarkable performance versus uniformly distributed random sampling.

Examples of this clustering algorithm applied to multiple types of modulations can be seen in Figures 4.6 through 4.14. These figures include 4-QAM, 16-QAM, 64-QAM, and 256-QAM, as well as QPSK, 8-PSK, 16-PSK, and 32-PSK modulations with SNR = 5, 10, 20, and 30 dB. Figure 4.15 shows the application of this algorithm to PAM modulations with SNR = 20 dB.
In high SNR (SNR = 30 dB and higher), the performance of the algorithm is almost perfect. In the case of 256-QAM modulation we see a few missing symbols in the constellation, which decreases when we double the number of samples from 512 to 1024. This is due to the fact that in the case of 256-QAM modulation and 512 samples, there are only two samples per symbol in average. However, even in the case of 512 samples and for 256-QAM, the shape of constellation is still clear. We will show later that the algorithm is fully able to classify the type of modulation even when a few symbols are missing as seen in the case of 256-QAM and using CDE would give us a large enough threshold to distinguish this modulation from
other modulations. This will be demonstrated in Chapter 5. As we move toward lower SNR (SNR = 5 dB and lower), we see that the cluster centers are deviating from the actual positions of constellation symbols.

In the case of PSK modulations, the degradation starts in higher SNRs compared to QAM modulations. This is due to the Euclidean distance of the symbols in PSK, which is much closer compared to QAM.

In Figure 4.15, for the case of PAM modulation, we observe a quadrature element in the signal. This is due to the simulation code in MATLAB for adding a complex white Gaussian noise. Normally, PAM signals do not contain any quadrature element and are purely real signals, but we observe quadrature elements due to noise. However, this does not affect the results of the simulation as we only take the in-phase component of the signal in this case.
Figure 4.3. Comparison of error rate for clustering algorithm with \( k \)-center initialization vs. uniformly distributed random initialization, for SNR = 5 dB.
Figure 4.4. Comparison of error rate for clustering algorithm with $k$-center initialization vs. uniformly distributed random initialization, for SNR = 15 dB.
Figure 4.5. Comparison of error rate for clustering algorithm with $k$-center initialization vs. uniformly distributed random initialization, for SNR = 30 dB.

It should be noted that, despite using only 512 samples of the signal, the algorithm yields accurate results even for higher order modulations such as 256-QAM. The missing cluster centers in the results from 256-QAM are due to the absence of samples representing those symbols. However, even for the symbols with only one or two samples, the algorithm is still able to find the representing symbol.

In Chapter 6, when we talk about using symbol distances after the clustering algorithm to determine the type of modulation scheme, we will explain how it can be used to classify higher order modulations with number of samples less than the number of symbols. For now we assume that $N \gg K$, where $N$ is the number of
samples and $K$ is the number of symbols in a modulation.\(^3\)

---

\(^3\)Optimizing the number of samples, or determining whether we can achieve good results when $N < K$ can be a possible research topic or future work.
Figure 4.7. Clustering algorithm performed on (a) 4-QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM signals with SNR = 30 dB and 1024 samples. The red dots show the center of each cluster.
Figure 4.8. Clustering algorithm performed on (a) QPSK, (b) 8-PSK, (c) 16-PSK, and (d) 32-PSK signals with SNR = 30 dB and 512 samples. The red dots show the center of each cluster.
Figure 4.9. Clustering algorithm performed on (a) 4-QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM signals with SNR = 20 dB and 512 samples. The red dots show the center of each cluster.
Figure 4.10. Clustering algorithm performed on (a) QPSK, (b) 8-PSK, (c) 16-PSK, and (d) 32-PSK signals with SNR = 20 dB and 512 samples. The red dots show the center of each cluster.
Figure 4.11. Clustering algorithm performed on (a) 4-QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM signals with SNR = 10 dB and 512 samples. The red dots show the center of each cluster.
Figure 4.12. Clustering algorithm performed on (a) QPSK, (b) 8-PSK, (c) 16-PSK, and (d) 32-PSK signals with SNR = 10 dB and 512 samples. The red dots show the center of each cluster.
Figure 4.13. Clustering algorithm performed on (a) 4-QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM signals with SNR = 5 dB and 512 samples. The red dots show the center of each cluster.
Figure 4.14. Clustering algorithm performed on (a) QPSK, (b) 8-PSK, (c) 16-PSK, and (d) 32-PSK signals with SNR = 5 dB and 512 samples. The red dots show the center of each cluster.
4.1.2 Issues With Single-Carrier Modulation Classification

In the performance of the clustering algorithm, there are a few issues that should be addressed before applying this method for real-life signals. To provide an example of these issues, as shown in Figures 4.6(d) and 4.13(d), there are some cases when one or more cluster centers are missing or erroneously misplaced due to noise or
any special symbol configuration in the modulation, which may cause an error in
the classification (e.g., giving a wrong number of cluster centers or detecting the
incorrect modulation type). We are considering these issues in this algorithm so
that it can also detect these conditions and reconfigure for these conditions.

Also, I and Q branches should have equal clocking to avoid the errors in forming
clusters of symbols. But we may be able to address some of these issues using
the same cluster-finding algorithm. We discuss them later in this chapter when
explaining OFDM feature extraction methods.

4.2 Multi-Carrier Modulation Classification

Passing the Gaussianity test indicates that the received signal has a Gaussian
distribution. However, we have to take note that this can be due to the presence
of plain additive white Gaussian noise in AWGN channels. It has been shown
that an OFDM signal is cyclostationary with period $T_s$\footnote{In cyclostationary process, the statistical properties are not time independent, but periodic
with time. In other words, if the mean and autocorrelation of a process $x(t)$ are periodic, then
the process is said to be a cyclostationary process.} [Bolcskei, 2001; Oner and
Jondral, 2004]. This means that the correlation function of the received signal

$$R_{rr}(t, \tau) = E\{r(t)r^*(t + \tau)\}$$ (4.15)

is periodic in $t$ for a given delay $\tau$. So, in the next step, a cyclostationarity test
is used to confirm if we indeed have an OFDM signal. If the test fails and no
cyclostationarity is detected then we can conclude that the incoming signal is not
OFDM but rather white Gaussian noise.

A by product of this process is the estimation of the OFDM symbol rate [Li
et al., 2006]. After this test, other processes are performed to extract parameters
such as cyclic prefix duration and number of subcarriers. These parameters are
required for demodulation of OFDM signal. Cyclic prefix is necessary to determine
the start of each symbol duration and number of subcarriers determines the number
of FFTs we need in the demodulator of the signal.
4.2.1 Cyclostationarity Test for Checking Validity of Gaussianity Tests

As stated before, a passed test of Gaussianity indicates Gaussian distribution in the incoming signal. However, a cyclostationarity test will be needed to distinguish between OFDM signal and AWGN.

The cyclic autocorrelation vector \( \hat{\mathbf{r}}_{\mathbf{r}^*} \) is given by [Oner and Jondral, 2004; Lundén et al., 2009]:

\[
\hat{\mathbf{r}}_{\mathbf{r}^*} = \begin{bmatrix}
\Re \{ \hat{R}_{\mathbf{r}^*}(\alpha, \tau_1) \}, & \ldots, & \Re \{ \hat{R}_{\mathbf{r}^*}(\alpha, \tau_N) \}, \\
\Im \{ \hat{R}_{\mathbf{r}^*}(\alpha, \tau_1) \}, & \ldots, & \Im \{ \hat{R}_{\mathbf{r}^*}(\alpha, \tau_N) \}
\end{bmatrix}. 
\] (4.16)

An estimate of the cyclic autocorrelation \( \hat{R}_{\mathbf{r}^*}(\alpha, \tau) \) may be obtained using \( M \) observations as

\[
\hat{R}_{\mathbf{r}^*}(\alpha, \tau) = \frac{1}{T_0} \sum_{t=0}^{T_0-1} r(t) r^*(t + \tau) e^{-j2\pi\alpha t} = R_{\mathbf{r}^*}(\alpha, \tau) + \varepsilon_{\mathbf{r}^*}(\alpha, \tau), 
\] (4.17)

where \( r(t) \) denotes the received complex valued signal, \( t \) is the discrete time index, \((*)\) denotes an optional complex conjugation, and \( \varepsilon_{\mathbf{r}^*}(\alpha, \tau) \) is the estimation of error. The notation covers both cyclic autocorrelation and conjugate cyclic autocorrelation with only one expression. It is assumed that \( r(t) \) has zero mean (in practice the mean can be estimated and subtracted from the signal). In addition, we assume the signal to be sufficiently oversampled. Oversampling at rate \( f_s \geq 2KB \), where \( K \) is the order of cyclostationarity and \( B \) is the monolateral signal bandwidth (i.e., \([-B, B]\)), eliminates aliasing in the cyclic frequency domain. We are considering the covariance of the signal to detect the cyclostationarity. Thus, the cyclostationarity in this work has an order of two. It follows this that the oversampling rate should be \( f_s \geq 4B \).

Similar to the equation above, the row vector of the true (asymptotic) value of the cyclic autocorrelation function \( R_{\mathbf{r}^*}(\alpha, \tau) \) is defined by

\[
\hat{\mathbf{r}}_{\mathbf{r}^*}(\alpha, \tau) = \mathbf{r}_{\mathbf{r}^*}(\alpha, \tau) + \varepsilon_{\mathbf{r}^*}(\alpha, \tau). 
\] (4.18)
Then, the test for the presence of second-order (conjugate) cyclostationarity at any of the cyclic frequencies of interest $\alpha$ simultaneously is formulated as:

$$
\mathcal{H}_0 : \hat{r}_{rr(*)}(\alpha, \tau) = \varepsilon_{rr(*)}(\alpha, \tau) \quad \text{signal not present,}
$$

$$
\mathcal{H}_1 : \hat{r}_{rr(*)}(\alpha, \tau) = r_{rr(*)}(\alpha, \tau) + \varepsilon_{rr(*)}(\alpha, \tau) \quad \text{signal present.}
$$

### 4.2.2 Estimation of Cyclic Prefix Duration

As mentioned above, the cyclostationarity test will give an estimation of OFDM symbol duration. OFDM symbol duration consists of two parts: the data duration and cyclic prefix duration. After we acquire symbol duration, estimation of the cyclic prefix duration will be important for ISI (Inter-Symbol Interference) elimination. For the cases of Rician and Rayleigh channels, there is multipath fading that may result in ISI. To eliminate any ISI without changing the orthogonality of its subcarriers, the last cyclic prefix of the useful signal is copied to the front of the symbol as shown in Figure 4.16. If we take $T_s$ to be the duration of one OFDM signal, we can write it as

$$
T_s = T_d + T_{cp},
$$

(4.19)

where $T_d$ is data duration and $T_{cp}$ is cyclic prefix duration. Because of copying of cyclic prefix, there is an extremum in the signal’s autocorrelation, which will indicate the duration of this cyclic prefix. So, a simple autocorrelation test will help to perform the task of estimation of cyclic prefix [Mobasseri, 2000].

Due to presence of this cyclic prefix, we can write the autocorrelation with an unknown delay $\tau$ as

$$
R_{rr}(n, \tau) = \begin{cases} 
\sigma_r^2 + \sigma_\omega^2, & \tau = 0 \\
\sigma_r^2 e^{-j2\pi \tau}, & \tau = T_d \\
0, & \text{otherwise}
\end{cases}
$$

(4.20)

where $\sigma_s^2 = E\{r(t)^2\}$ and $\sigma_\omega^2 = E\{n(t)^2\}$ represent the energy of useful signal and noise, respectively. By choosing a careful threshold, we can determine the length of cyclic prefix.
Figure 4.16. Cyclic prefix being copied for eliminating ISI.

Figure 4.17. The result of autocorrelation test. The peak shows where the cyclic prefix ends and the data begins in the duration of one symbol.

Figure 4.17 shows the autocorrelation of an OFDM signal with a symbol consisting of 64 data samples and 16 prefix samples, where cyclic prefix is the repeat of the end of the symbol at the beginning as seen in Figure 4.16.

4.2.3 Estimation of Number of Subcarriers

In Li et al. [2006], we see a detailed approach towards the estimation of the number of subcarriers. To estimate the number of carriers, $N$, we use a bank of Fast Fourier Transforms (FFTs). We assume that the number of subcarriers is a power
of two, since OFDM signals are made using inverse FFTs. This FFT bank consists of different lengths of FFTs based on powers of two and uses the concept of Gaussianity of the OFDM signals one more time to determine the number of subcarriers.

It utilizes the fact that, if the output in one of the FFT branches is perfectly demodulated, then it will have only useful data and will no longer possess a Gaussian distribution. On the other hand, all the other branches will still show Gaussian property. By increasing the number of OFDM symbols processed in this FFT bank, a more accurate result can be obtained, but accuracy would be a trade off with an increase in computation time to make the decision.

Assume the transmitter inverse discrete Fourier transform (IDFT) size is $N$ and the classifier discrete Fourier transform (DFT) size $\tilde{N}$ satisfy $\tilde{N} = MN$, where $M \geq 1$ is a positive integer. The input signal to the classifier DFT is

$$y_n^m = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_k^m e^{j2\pi kn/N},$$

(4.21)

where $X_k^m$ is the $k$th data symbol of the $m$th transmitted OFDM symbol and $y_n^m$ is the $n$th IDFT output symbol of the $m$th transmitted OFDM symbol. The classifier performs an $\tilde{N}$-point DFT hence the $k$th entry of the DFT output is given by:

$$Y_k = \frac{1}{\sqrt{\tilde{N}}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} y_n^m e^{-j\frac{2\pi k}{\tilde{N}}(mN+n)}.$$  

(4.22)

When $\frac{k}{M} = l$, Equation (4.22) can be simplified as:

$$Y_k = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} X_k^m.$$  

(4.23)

The physical meaning of Equation (4.23) is that, if $k$ is a multiple of $M$, then $Y_k$, the $k$th output of $\tilde{N}$-point FFT, is the summation of $M$ original data symbols. Those data symbols come from the $\frac{k}{M}$th subcarrier of $M$ transmitted OFDM symbols. Since $M$ may not be a large number, $Y_k$ shows little Normality.

Therefore, the procedure will be as follows: with the incoming signal, we initiate a $\tilde{N}$-point FFT operation. The initial value of $\tilde{N}$ is set larger than the possible
maximal value of the transmitter IFFT size $N$. Then, we test the output of the FFT for Normality. If strong Normality is shown, which means $\tilde{N} \gg N$, we divide $\tilde{N}$ by 2 and apply the new $\tilde{N}$-point FFT. This cycle is repeated until the Normality test fails, which implies the fact that $\tilde{N} = N$. Therefore, the number of subcarriers is obtained. If no result is achieved by the end, this repetition continues until $\tilde{N} = 2$ indicates an error in this step of determining the number of subcarriers, or in previous step of determining the validity of the Gaussian tests.

In Shi et al. [2007], the estimation of the number of subcarriers is made simpler by assuming that we know exactly the symbol and the bit duration. The concept of the blind estimation is used here, wherein this value is corrected at various predictor-corrector steps and estimated using Blind Estimation Theory.

The number of subcarriers is given by the simple ratio of $N = \lceil T_s/T_b \rceil$ where the estimate of $T_s$ will be iteratively estimated and can be viewed as a summation of the autocorrelation of the signal.

4.2.4 OFDM Synchronization

Before our algorithm can extract OFDM features and demodulate the subcarriers, it must perform at least two synchronization tasks:

1. Determine where the symbol boundaries are and what the optimal timing intervals are to minimize the effects of ICI (Inter-Carrier Interference) and ISI; and

2. Estimate and correct for the carrier frequency offset of the received signal.

If we assume that the OFDM signals are generated by IDFT and demodulated by DFT, we can get a lower bound for the SNR [Xiong, 2006]. The following assumptions are made for the channel. It is modeled by a complex transfer function $H_k$ for the $k^{th}$ subcarrier and a frequency offset $\epsilon$ for all subcarriers, where $\epsilon$ is relative to the date symbol rate $R_s = 1/T_s$. It is also a slow-fading channel. Based on this model, the lower bound for the SNR will be as follows

$$\text{SNR} \geq \frac{E_c}{N_0} \left[ \frac{\sin(\pi \epsilon)}{\pi \epsilon} \right]^2 \frac{1}{1 + 0.5947 \frac{E_c}{N_0} [\sin(\pi \epsilon)]^2}, \quad |\epsilon| \leq 0.5 , \quad (4.24)$$
where $E_c$ is the average received symbol energy of the individual subcarriers, which include the channel gains, and $N_0$ is the AWGN power spectral density in the bandpass transmission channel.

With the previous steps in our algorithm, we have demonstrated how to find the OFDM symbol boundaries. Here we also propose a method to correct frequency offsets of the signal. First however, we overview the effects of frequency and timing offsets on the OFDM signal.

The SNR at the input of the decision-making part of the demodulator is given by Pollet et al. [1995]:

$$\text{SNR} = \frac{E_0^2}{N_0/E_s + V_0},$$

(4.25)

where $E_0^2$ is the power of the useful component of the signal, $N_0/E_s$ shows the variance of the thermal noise contribution, and $V_0$ denotes the variance of the other noise terms.

Comparing Equation (4.25) with $\text{SNR} = E_s/N_0$ in the absence of carrier phase offsets, the degradation in dB is defined as

$$D_{\text{deg}} = -10 \log_{10} \left( \frac{E_0^2}{1 + V_0 E_s N_0} \right),$$

(4.26)

$$= -10 \log_{10} (E_0^2) + 10 \log_{10} \left( 1 + V_0 \frac{E_s}{N_0} \right).$$

(4.27)

When the offsets are small, $D_{\text{deg}}$ is approximated by

$$D_{\text{deg}} \approx \frac{10}{\ln 10} \left( 1 - E_0^2 + V_0 \frac{E_s}{N_0} \right).$$

(4.28)

The following sections present the equations for degradation due to each offset element that contribute to degradation in Equation (4.28). The details of these calculations are presented in Pollet et al. [1995].

### 4.2.5 Sensitivity to Phase Noise

In Nee and Prasad [2000], the sensitivity of OFDM signal to phase noise has been discussed and two effects for phase noise have been introduced. First, it introduces a random phase variation that is common to all subcarriers. When the oscillator
line-width is much smaller than the OFDM symbol rate, which is usually the case, then common phase error is strongly correlated from symbol to symbol and tracking techniques or differential detection is commonly used to minimize the effect. The second, and perhaps disturbing effect, is that it introduces ICI because the subcarriers are no longer spaced at exactly $1/T_s$ in the frequency domain. The amount of ICI can be calculated and translated into a degradation in SNR that is given as

$$D_p \approx \begin{cases} \frac{11}{6 \ln 10} (4\pi N \frac{\beta}{R}) \frac{E_s}{N_0} & \text{OFDM} \\ \frac{11}{6 \ln 10} (4\pi \frac{\beta}{R}) \frac{E_s}{N_0} & \text{SC} \end{cases}$$

(4.29)

where $\beta$ is the $-3$ dB one-sided bandwidth of the power density spectrum of the carrier, $N$ is the number of subcarriers, $R$ is the fixed total symbol rate, with $R = N/T_s = NR_s$ in the case of OFDM signals, and $R = 1/T_s = R_s$ in the case of single-carrier (SC) signals, and $E_s = \mathbb{E}\{|a_k|^2\}$ is the average symbol energy for subcarrier $k$. The phase noise degradation is proportional to $\beta T_s$, which is the ratio of the line-width and subcarrier spacing $1/T_s$. $E_s/N_0$ is the same as $E_c/N_0$, except that $E_c$ includes the channel gain. $N_0/2$ is the AWGN power spectral density in the bandpass transmission channel. Figure 4.18 shows this degradation in the case of QPSK, 16-QAM, and 64-QAM modulations.

### 4.2.6 Sensitivity to Frequency Offset

All OFDM subcarriers are orthogonal if they all have a different integer number of cycles within the FFT interval. If there is a frequency offset, then the number of cycles in the FFT interval is not an integer, resulting in ICI occurring after the FFT.

For a fixed total symbol rate $R$, the degradation (dB) due to frequency offset is

$$D_f \approx \begin{cases} \frac{10}{\ln 10} \frac{1}{3} (\pi N \frac{\Delta F}{R})^2 \frac{E_s}{N_0} & \text{OFDM} \\ \frac{10}{\ln 10} \frac{1}{3} (\pi \frac{\Delta F}{R})^2 & \text{SC} \end{cases}$$

(4.30)
4.2.7 Sensitivity to Timing Errors

The effect of timing offset is derived in Xiong [2006]. This effect within the range of the ISI-free part of the cyclic prefix is to cause a phase rotation of $\exp(-j2\pi\frac{k\tau}{N})$ on the $k$th subcarrier, where $\tau \in \{1, \ldots, N_g - \tau_{\text{max}}\}$ is the timing error, in which $N_g$ is the guard interval and $\tau_{\text{max}}$ is the maximum length of ISI of the previous symbol into the guard interval of the current symbol. If the timing offset is in the range given, orthogonality between subcarriers is preserved and the offset introduces a phase rotation. This rotation is linearly proportional to the subcarrier frequency. If the offset is out of the given range, the orthogonality will be destroyed and severe symbol errors occur.

Figure 4.18. Phase sensitivity on SNR degradation for 64-QAM ($E_s/N_0 = 19$ dB), 16-QAM ($E_s/N_0 = 14.5$ dB), QPSK ($E_s/N_0 = 10.5$ dB).
4.2.8 Using Clustering Algorithm for Correcting Offsets in OFDM Signal

The effects of offsets on an OFDM signal are shown in Figures 4.20 through 4.25. In each figure, each offset is applied without the effect of any of the other offsets. It is seen that in each case the correct centers for clusters are determined. This provides us with a potential reference point to measure and correct these offsets, when only one is present.

Figures 4.20 through 4.22 show the effect of carrier frequency offset, frame time offset, and sampling clock offset, when each of them are 5%, 10%, and 15% off respectively. As it can be seen, the effect of frame time offset degradation on I–Q diagram is a circular phase rotation. The algorithm is able to detect the correct location of the original symbol. By calculating the distance of the farthest point in each cluster from the cluster center, we can measure this offset and correct it. Figure 4.26 shows the results from the algorithm for correcting frame time offset. It is seen that the results obtained from our algorithm are linearly dependent on frequency offset, making it possible to correct this offset. However, carrier frequency offset and sampling clock offset do not show a traceable pattern for different amounts of offsets, but cluster centers are still correctly detected.

Figures 4.23 through 4.25 show the effect of I–Q gain imbalance, phase imbalance, and differential time, when each of them are imbalanced by 10%, 20%, and 30%.

For phase imbalance, this degradation appears as a rotation of all the symbols. This can also be measured by the degree of rotation of centroids compared to vertical or horizontal reference lines. For instance, if we take the angle shown in Figure 4.19 and compare it to the I–Q phase imbalance, it can be seen in Figure 4.27 that just like previous case, they have an almost linear relationship that can be used as a way of measuring and correcting the phase imbalance between in-phase and quadrature components.

There is another feature on I–Q diagrams that is common between I–Q gain imbalance and phase imbalance and can be used to correct these imbalances. For instance, in the example of I–Q gain imbalance, we observe that as the imbalance increases, a 16-QAM modulation of each symbol forms around that symbol and
Figure 4.19. Measuring the angle to calculate the I–Q phase imbalance in a 16-QAM modulation.

it grows. By measuring this effect in each symbol, we can similarly calculate and correct this imbalance.

In the cases of sampling clock offset and I–Q differential time, we would need more advanced methods to correct these degradations using these methods. However, it is seen that the clustering algorithm gives us a solid starting point by detecting the cluster centroids correctly.

4.3 Summary

In this chapter, we described the details of our algorithm. We showed that, by applying a greedy $k$-center algorithm on I–Q diagrams, we can achieve high success rate of classification for QAM and PSK algorithms, as well as for any modulation scheme with a distinct I–Q diagram such as PAM.

We presented a number of initial results to support our method, which showed that this algorithm can perfectly detect the cluster centers for the symbols in a modulation. Furthermore, to be able to quantize this performance and to come
up with a scale that would allow us to differentiate between various modulation schemes, we defined a term that we called “Relative Error” or Cumulative Deviation Error (CDE). By defining this value, we were able to demonstrate two of the main contributions of this project. First, we were able to show that using $k$-center algorithm in conjunction with $k$-means clustering greatly improves the performance of finding cluster centers, especially in the case of higher SNRs. Second, we used CDE as a measurable parameter to use in our classifier for making the decisions.

With this step accomplished, the only remaining part of the algorithm is OFDM feature extraction. These include the extraction of synchronization parameters of the signal, such as symbol duration, cyclic prefix duration, and data duration. Furthermore, we need to know the number of subcarriers in order to extract the date. These steps have been previously developed in the literature. By implementing them in our algorithm, we showed that they can be paired with the rest of the MC algorithm to complete our MC algorithm.

One of the important obstacles in OFDM feature extraction is the signal synchronization and timing and frequency offset correction of the OFDM signal. We also applied our clustering algorithm to I–Q diagrams of an OFDM signal to show that this algorithm can be used to correct these time and frequency offsets. We demonstrated that for the cases of frame time offset and I–Q phase imbalance, there are features extracted from I–Q diagram that are linearly dependent on the offsets and imbalances. These features can be measured using our clustering algorithm and can be used to correct these offsets and imbalances.
Figure 4.20. Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 5% carrier frequency offset, (c) with 5% frame time offset, and (d) with 5% sampling clock offset. The red stars show the center of each cluster.
Figure 4.21. Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 10% carrier frequency offset, (c) with 10% frame time offset, and (d) with 10% sampling clock offset. The red stars show the center of each cluster.
Figure 4.22. Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 15% carrier frequency offset, (c) with 15% frame time offset, and (d) with 15% sampling clock offset. The red stars show the center of each cluster.
Figure 4.23. Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 10% I–Q gain imbalance, (c) with 10% I–Q phase imbalance, and (d) with 10% I–Q differential time. The red stars show the center of each cluster.
Figure 4.24. Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 20% I–Q gain imbalance, (c) with 20% I–Q phase imbalance, and (d) with 20% I–Q differential time. The red stars show the center of each cluster.
Figure 4.25. Clustering algorithm performed on an OFDM signal: (a) Original signal, (b) with 30% I–Q gain imbalance, (c) with 30% I–Q phase imbalance, and (d) with 30% I–Q differential time. The red stars show the center of each cluster.
Figure 4.26. Correcting the frame time offset in an OFDM signal (16-QAM): measured offset by classifying algorithm versus actual offset.
Calculating I−Q phase imbalance using the proposed algorithm in 16−QAM

**Figure 4.27.** Correcting the I–Q phase imbalance in an OFDM signal (16-QAM): measured imbalance by calculating the angle between the axis of 16-QAM signal with x axis using classifying algorithm. The unit on the x axis shows the percentage of the I–Q phase imbalance present.
Chapter 5

Performance Test for New MC Method

5.1 Introduction

The signals that we have considered herein for modulation classification include the widely used single-carrier digital modulations $M$-QAM, $M$-PAM, and $M$-PSK. For multi-carrier modulations, the signal considered in simulations is the OFDM signal. OFDM is currently employed in many wireless communication systems. Some of these systems include 3GPP Long Term Evolution (LTE), IEEE 802.11a/g/n Wireless Local Area Networks (WLAN), Digital Video Broadcasting (DVB) standards DVB–T and DVB–H, IEEE 802.16, and WiMax Wireless Metropolitan Area Networks (MAN).

In this chapter, first we will briefly talk about one of the methods we have applied to implement this algorithm on hardware. Then, we will talk more thoroughly about the results we have got from our simulations and what they mean.

5.2 Using Simulink for Hardware Implementation

For simulating an OFDM signal, we have written code in MATLAB along with models available in Simulink to create OFDM–16QAM, –BPSK, –QPSK, and
–8PSK signals. The block diagram of the Simulink model is shown in Figure 5.1. This model can be converted to VHDL or Verilog using Xilinx Simulink Toolbox and Xilinx ISE software and be coded into Xilinx FPGA chips on some of the available SDR boards. The OFDM modulator and demodulator use IFFT and FFT processes for performing their tasks. The transmitted signal and the received signal for the model are depicted in Figure 5.2 and their I–Q diagrams in Figure 5.3. The simulation first uses a block to convert data from series to parallel format. The next blocks ensure that the frequency domain representation has FFT Hermitian symmetry. Hermitian symmetry means that the following relation is ensured between carriers:

\[ g_i = g^*_n - i, \]  

(5.1)

where \( n \) is the total number of carriers and \( * \) indicates the complex conjugate.

After this step, the subcarriers will be dealt with in the same manner as single-carrier modulations. The I–Q diagrams of the transmitted and received signals of an OFDM–16QAM are shown in Figure 5.3. The signals from this simulation are used to study the normality tests as well as the performance of our algorithm under different channel characteristics.

5.3 Clustering Algorithms in I–Q Diagrams

We have performed various tests to determine how efficient our clustering algorithm is compared with other algorithms and how we can optimize the algorithm with respect to number of samples and number of transmitted bits.

Figures 5.4 through 5.9 show the relative error with respect to number of transmitted samples in 4-QAM, 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations in the presence of 20-dB SNR. In all these figures we observe that the CDE decreases sharply until it reaches a certain threshold after which it still continues to decline but not as sharply as before. We can conclude that until a certain number of samples, adding any new sample has a significant effect on the accuracy of our algorithm and in finding the exact cluster center. However, after this threshold is passed the new added samples won’t have the same effect. Since

\(^{1}\)All of the simulations in this thesis are performed using a system equipped with Intel Core i7 processor, 12 GB memory, Windows7 64-bit operating system, and MATLAB R2009a x64.
our aim is also to optimize the algorithm and find the fewest number of samples possible, finding this threshold level of samples can be very helpful.
Figure 5.2. (a) Spectrum of transmitted signal and (b) spectrum of received signal.

Figure 5.3. I–Q diagrams of (a) transmitted and (b) received signals for OFDM–16QAM modulation.
### Average error power in finding cluster centers in a 4-QAM signal

<table>
<thead>
<tr>
<th>Number of samples</th>
<th>Relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10⁻¹</td>
</tr>
<tr>
<td>2</td>
<td>10⁻²</td>
</tr>
<tr>
<td>3</td>
<td>10⁻¹</td>
</tr>
</tbody>
</table>

**Figure 5.4.** Relative error with respect to number of samples in a 4-QAM modulation with SNR = 20 dB.
Average error power in finding cluster centers in a 8-QAM signal

Number of samples
Relative error

Figure 5.5. Relative error with respect to number of samples in a 8-QAM modulation with SNR = 20 dB.
Figure 5.6. Relative error with respect to number of samples in a 16-QAM modulation with SNR = 20 dB.
<table>
<thead>
<tr>
<th>Number of samples</th>
<th>Relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

**Figure 5.7.** Relative error with respect to number of samples in a 64-QAM modulation with SNR = 20 dB.
Figure 5.8. Relative error with respect to number of samples in a 8-PSK modulation with SNR = 20 dB.
Figure 5.9. Relative error with respect to number of samples in a 16-PSK modulation with SNR = 20 dB.

Figures 5.10 through 5.13 show the relative error with respect to number of transmitted bits in 4-QAM, 8-QAM, 64-QAM, and 8-PSK modulations in the presence of 20-dB SNR. Comparing these figures to each other, we do not observe any correlated distinction between these graphs. However, this would be logical. Since we are considering the signal before the demodulator, we would not have any knowledge of the bits per symbol.
Figure 5.10. Relative error with respect to number of bits in a 4-QAM modulation with SNR = 20 dB.
Figure 5.11. Relative error with respect to number of bits in a 8-QAM modulation with SNR = 20 dB.
Figure 5.12. Relative error with respect to number of bits in a 64-QAM modulation with SNR = 20 dB.
Figure 5.13. Relative error with respect to number of bits in a 8-PSK modulation with SNR = 20 dB.
Figures 5.14 and 5.15 show the relative error that indicates the deviation of the found cluster centers from the actual modulation states in the studied modulations with respect to SNR. The number of samples in all of these simulations is 512.

Figure 5.14. Relative error shows the deviation of found cluster centers from the actual cluster centers in 4-QAM and 8-QAM signals with respect to SNR.
Figure 5.15. Relative error shows the deviation of found cluster centers from the actual cluster centers in 8-PSK signal with respect to SNR.

Figures 5.16 through 5.19 show the relative error (CDE) for a received 8-PSK signal, when compared to 8-PSK modulation scheme in our bank of modulations correctly and when compared to QPSK, 8-QAM, 16-QAM, and 16-PSK. 8-PSK is chosen as an example for this purpose, to compare it both to higher and lower order modulation schemes. In both cases we see that relative error gives us a distinct threshold level for choosing the correct modulation scheme. This distinction works well for SNR levels as low as 3 dB. By choosing a carefully set threshold in a way that the CDE derived from correct modulation scheme would be smaller and the CDE from the incorrect modulation schemes would be greater than this threshold,
we can device our classifier in a way that it would choose the correct modulation. Figures 5.20 through 5.23 show an even better distinction when we use $\varepsilon_{CDE}^2$.

![Figure 5.16](image.png)

**Figure 5.16.** Comparison of relative error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 20 dB.
Comparison of average error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations, with SNR = 10 dB.

**Figure 5.17.** Comparison of relative error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 10 dB.
Comparison of average error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations, with SNR = 5 dB.

Figure 5.18. Comparison of relative error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 5 dB.
Figure 5.19. Comparison of relative error power for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 3 dB.
Figure 5.20. Comparison of $\varepsilon^2_{\text{CDE}}$ for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 20 dB.
Comparison of $\epsilon^2$ for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations, with SNR = 10 dB.

**Figure 5.21.** Comparison of $\epsilon^2_{CDE}$ for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 10 dB.
Figure 5.22. Comparison of $\varepsilon^2_{\text{CDE}}$ for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations with SNR = 5 dB.
Comparison of $\varepsilon^2$ for a received 8-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations, with SNR = 3 dB.

Figures 5.25 through 5.29 give further examples on the performance of this algorithm. They include applying the classifier to received 16-PSK, 16-QAM, and 64-QAM modulated signals. In these cases also there is a very good distinction between the correct and incorrect modulation schemes in low SNR levels, clear enough to enable classification.
Figure 5.24. Comparison of $\varepsilon_{\text{CDE}}^2$ for a received 16-PSK modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 20 dB.
Comparison of $\varepsilon^2$ for a received 16-PSK modulated signal when compared vs. 8-PSK, QPSK, 8-QAM, 16-QAM, and 16-PSK modulations, with SNR = 3 dB.

**Figure 5.25.** Comparison of $\varepsilon_C^2$ for a received 16-PSK modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 3 dB.
Comparison of $\varepsilon^2$ for a received 16-QAM modulated signal when compared vs. 16-QAM, 8-QAM, 64-QAM, 8-PSK, and 16-PSK modulations, with SNR = 20 dB.

**Figure 5.26.** Comparison of $\varepsilon^2_{CDE}$ for a received 16-QAM modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 20 dB.
Comparison of $\varepsilon^2$ for a received 16-QAM modulated signal when compared vs. 16-QAM, 8-QAM, 64-QAM, 8-PSK, and 16-PSK modulations, with SNR = 3 dB.

<table>
<thead>
<tr>
<th>Modulation</th>
<th>Number of Samples</th>
<th>$\varepsilon^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-QAM</td>
<td>$10^{-2}$</td>
<td></td>
</tr>
<tr>
<td>64-QAM</td>
<td>$10^{-1}$</td>
<td></td>
</tr>
<tr>
<td>8-QAM</td>
<td>$10^{0}$</td>
<td></td>
</tr>
<tr>
<td>8-PSK</td>
<td>$10^{1}$</td>
<td></td>
</tr>
<tr>
<td>16-PSK</td>
<td>$10^{2}$</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5.27.** Comparison of $\varepsilon^2_{\text{CDE}}$ for a received 16-QAM modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 3 dB.
Comparison of $\varepsilon^2$ for a received 64-QAM modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations, with SNR = 20 dB.

**Figure 5.28.** Comparison of $\varepsilon^2_{CDE}$ for a received 64-QAM modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 20 dB.
Comparison of $\varepsilon^2$ for a received 64-QAM modulated signal when compared vs. 64-QAM, 8-QAM, 16-QAM, 8-PSK and 16-PSK modulations, with SNR = 3 dB.

![Graph showing comparison of $\varepsilon^2$ for different modulations.](image)

**Figure 5.29.** Comparison of $\varepsilon^2_{\text{CDE}}$ for a received 64-QAM modulated signal when compared vs. 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulations with SNR = 3 dB.

After examining the possible options for the threshold in Figures 5.26 through 5.29, we simulate the classifier and test it for 4-QAM, 8-QAM, 16-QAM, 64-QAM, 8-PSK, 16-PSK modulated signals. Tables 5.1 through 5.4 show some of the results obtained in the cases, where the received signal is modulated as 16-QAM, 64-QAM, 8-PSK, and 16-PSK. The results are from 10,000 trials of the simulation.
Table 5.1. Classifier results, when the input signal is 16-QAM modulated passed through an AWGN channel (in percentage).

<table>
<thead>
<tr>
<th>SNR</th>
<th>4-QAM</th>
<th>8-QAM</th>
<th>16-QAM</th>
<th>64-QAM</th>
<th>8-PSK</th>
<th>16-PSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>99.4%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>96%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>88%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.2. Classifier results, when the input signal is 64-QAM modulated passed through an AWGN channel (in percentage).

<table>
<thead>
<tr>
<th>SNR</th>
<th>4-QAM</th>
<th>8-QAM</th>
<th>16-QAM</th>
<th>64-QAM</th>
<th>8-PSK</th>
<th>16-PSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>3.3%</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>4.1%</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60.5%</td>
<td>2.2%</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60.5%</td>
<td>2.2%</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.3. Classifier results, when the input signal is 8-PSK modulated passed through an AWGN channel (in percentage).

<table>
<thead>
<tr>
<th>SNR</th>
<th>4-QAM</th>
<th>8-QAM</th>
<th>16-QAM</th>
<th>64-QAM</th>
<th>8-PSK</th>
<th>16-PSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99.9%</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>87%</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6.7%</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.4. Classifier results, when the input signal is 16-PSK modulated passed through an AWGN channel (in percentage).

<table>
<thead>
<tr>
<th>SNR</th>
<th>4-QAM</th>
<th>8-QAM</th>
<th>16-QAM</th>
<th>64-QAM</th>
<th>8-PSK</th>
<th>16-PSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>89.5%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>43.4%</td>
</tr>
</tbody>
</table>

At this point, we have evaluated the performance of our modulation classification algorithm, using various simulations. Now, we can compare this method to other classifiers. To be able to perform this the logical criteria are the complexity and duration of calculations (how fast they can classify different modulations) and their performance in low SNRs (how accurate they can classify). However, as Dobre et al. [2007] mentions, it turns out that performance comparison of published classifiers is not straightforward. This is due to a number of reasons. For example, some of the classifiers are specifically designed to handle specific unknown parameters and they have considered different types of modulations.

However, to take our shot at comparing our MC with those in the literature, we take Dobre et al. [2007] as the base reference point and to be able to compare them to our results, we assume the same parameters in our simulations.

They have examined ALRT, quasi-ALRT, HLRT and cumulant-based algorithms by choosing QPSK and BPSK as candidate modulations. They have defined the following parameters:

- $P_{cc}$ percentage of correct classification,
- 1000 Monte Carlo trials,
- $N = 100$ the number of symbols, and
- rectangular pulse shape.
It must be noted that the likelihood ratio tests from high complexity to low complexity are ALRT, quasi-ALRT, GLRT, HLRT, and quasi-HLRT. We assume the ideal scenario for these simulations. Table 5.5 shows a comparison of our results with those presented in Dobre et al. [2007] for classification of BPSK versus QPSK modulations.

Table 5.5. Comparing performance and details of our method with previously developed methods for BPSK vs. QPSK.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>SNR (dB)</th>
<th>$P_{cc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC method in this work</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td>MC method in this work</td>
<td>0</td>
<td>94.2%</td>
</tr>
<tr>
<td>MC method in this work</td>
<td>−3</td>
<td>47.1%</td>
</tr>
<tr>
<td>ALRT, $L = 1$, $\eta_A = 1$</td>
<td>−3</td>
<td>97.5%</td>
</tr>
<tr>
<td>ALRT, $L = 2$, $\eta_A = 1$</td>
<td>−6</td>
<td>97.5%</td>
</tr>
<tr>
<td>quasi-ALRT, $M = 2$</td>
<td>−2</td>
<td>96%</td>
</tr>
<tr>
<td>HLRT, threshold = 1</td>
<td>−2</td>
<td>96.8%</td>
</tr>
<tr>
<td>cumulant-based, $N_{mod} = 2$, $\mu_H = 1$</td>
<td>4</td>
<td>96%</td>
</tr>
<tr>
<td>cumulant-based, $N_{mod} = 2$, $\mu_H = 1$</td>
<td>6</td>
<td>96%</td>
</tr>
<tr>
<td>quasi-ALRT, $M = 2$, timing offset = 0.15</td>
<td>11</td>
<td>96%</td>
</tr>
</tbody>
</table>

As can be seen, ALRT-based classifiers still perform better in this case but they are very sensitive to unknown carrier phase offsets.

Now, we compare our results with these classifiers for QAM modulates signals. We choose classification of 16-QAM versus 8-QAM and compare them with results from the same article. The results are compared in Table 5.6. The same number of symbols as the previous table was used for this comparison. As it can be seen, in this case the algorithm performs much better compared to other algorithms. It shows a 6-dB improvement over ALRT and 8-dB over HLRT algorithms.

It was not possible to directly compare the complexity of different algorithms with the algorithm presented in this work. However, to give an idea on how fast this algorithm can perform, we used tic and toc functions in our code with performing the simulations in this comparison. For 1000 trials of the simulation, which included generation of the modulated symbols, adding noise and classifying them,
Table 5.6. Comparing performance and details of our method with previously developed methods for 8-QAM vs. 16-QAM.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>SNR (dB)</th>
<th>$P_{\text{ex}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC method in this work</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td>MC method in this work</td>
<td>0</td>
<td>97.7%</td>
</tr>
<tr>
<td>MC method in this work</td>
<td>−3</td>
<td>34.3%</td>
</tr>
<tr>
<td>ALRT, $L = 1$, $\eta_A = 1$</td>
<td>7</td>
<td>99%</td>
</tr>
<tr>
<td>HLRT with $\mu_H$ not specified</td>
<td>9</td>
<td>99%</td>
</tr>
<tr>
<td>Cumulant-based, $N_{\text{mod}} = 2$, $\mu_H = -0.68$</td>
<td>9</td>
<td>99%</td>
</tr>
<tr>
<td>quasi-HLRT, threshold = 1</td>
<td>19</td>
<td>99%</td>
</tr>
<tr>
<td>quasi-ALRT</td>
<td>30</td>
<td>88%</td>
</tr>
</tbody>
</table>

It took 10.47 seconds for QAM modulations and 8.31 seconds for PSK simulations.

In the next section, we will discuss the complexity of our algorithm.

5.4 On the Complexity of the Algorithm

As mentioned in the introduction of this dissertation, it is clear that a modulation classifier should have the following characteristics:

- Provide a high probability of successful classification in a short observation interval and for a wide range of SNR,
- Recognize many different modulations in diverse propagation environments,
- Function in real time, and
- Have low computational complexity.

From the above characteristics of a modulation classifier, it can be seen that there are two aspects to examine to determine the complexity of an algorithm. The first is how well it works in real-time and, second, how computationally complex it is. Although these questions are interrelated in nature, the method to deal with them and the results can be completely independent of one another.
To start the evaluation and derive the complexity, we have to divide the algorithm into sub-algorithms, for end of which we can determine the complexity. If we take another look at Figure 1.3, we observe that the algorithm consists of three parts in the worst case scenario, which is the case of a received multi-carrier signal. These three parts consist of the Gaussianity test, the OFDM feature extraction, and the clustering algorithm on I–Q diagrams.

To address how well the algorithm deals with real-time scenarios, we have performed a number of simulations to calculate the processing time. The results for these simulations are presented in Chapter 3 and at the end of Section 5.3. However, to have an idea on the degree of complexity of the algorithm, we use the big $O$ notation, which gives the asymptotic running time for an indefinitely large input.

First, we consider the Gaussianity test. In Chapter 3 we concluded that the most appropriate tests for our purpose would be the Shapiro–Wilk test and the Cramer–von Mises test. In the Shapiro–Wilk test, the MATLAB function `polycal` is used. This function is based on Horner’s method of polynomial evaluation [Horner, 1819]. Based on this and the MATLAB code for Shapiro–Wilk test, we can conclude that the test is $O(N)$.

In the case of Cramer–von Mises test, the MATLAB function `interp1` is used, which is the function for linear interpolation. This function is also $O(N)$. Thus, we can conclude that the first stage of the algorithm is $O(N)$.

The second stage, OFDM feature extraction, consists of several steps. In this part, the most computationally complex process is the autocorrelation process. By itself, the autocorrelation function is $O(N^2)$. However, there are several efficient algorithms that can bring down the order of calculation to $O(N \log N)$.

Finally, the last step is the clustering algorithms of $k$-center and $k$-means. The greedy $k$-center algorithm that is used here, as an approximation of $k$-center algorithm is $O(\log N)$ [Archer, 2001; Li Gørtz and Wirth, 2006]. In the case of $k$-means there are several different approximations for this algorithm. The most common algorithm that is used for $k$-means and is also implemented in MATLAB is also called Lloyd’s algorithm [Lloyd, 1982]. The algorithm is usually very fast, but there is no guarantee that it will converge to a global minimum, and the result may depend on the method of initialization. It can become very slow for some initial
values and theoretically can take exponential time $2^{\Omega(\sqrt{N})}$ to converge [Arthur and Vassilvitskii, 2006b]. However, using the $k$-center algorithm for initialization guarantees that the algorithm converges quickly. Thus, we can conclude for this stage that the approximation is $\mathcal{O}(\log N)$.

After considering all of these stages, the stage with the highest complexity will determine the upper limit for the order of complexity of this algorithm. As can be seen from previous arguments, the second stage with a complexity order of $\mathcal{O}(N \log N)$ has the higher bound. Thus, it can be concluded that this is the order of complexity of this MC algorithm. However, it must also be noted that in modern computers, the complexity expressed by big $\mathcal{O}$ notation can be irrelevant. It is more important how long those operations take and what combinations of instructions can be processed simultaneously by the cpu. Also tremendously important in computing time is the ability of the algorithm to fit into cache. An algorithm that takes $\mathcal{O}(N)$ running time in theory can end up taking much longer than a different algorithm that takes $\mathcal{O}(N^2)$ to compute the same result, if the $\mathcal{O}(N^2)$ operates in cache and the $\mathcal{O}(N)$ trashes the cache badly.

5.5 Summary

In this chapter, we presented further results from our clustering algorithms. We also studied initial results for hardware implementation of the complete MC algorithm.

In the results presented in this chapter, we studied the performance of the clustering algorithm with respect to number of samples, number of bits, and noise for various PSK and QAM modulations. We also studied the error rate versus number of samples in our simulations to determine the minimum number of samples to achieve a reasonable error rate. For our simulations, we chose 512 samples and showed that, for SNR as low as 5 dB, the algorithm can still achieve error rate lower than 1%. We also defined a term as “relative error” to use as a discrimination parameter between modulation schemes and set a threshold to classify them. These results showed that for SNR as low as 5 dB this algorithm can return reliable results, when discriminating between 4-QAM, 8-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK. It was demonstrated that for lower SNRs, it can still be used to
set apart different families of modulations or used to classify limited numbers of modulations.

We also compared the performance of this algorithm with previous methods. In order to do this, we compared the performance of different algorithms in classifying BPSK/QPSK signals as well as 8-QAM/16-QAM signals. As shown in Tables 5.5 and 5.6, the performance of this algorithm is comparable to the best published classifiers in classifying the class of QAM or PSK modulation schemes and is superior in performance in being able to classify both QAM and PSK modulations with high accuracy in low SNRs.

Finally, we analyzed the complexity of our algorithm. For this purpose, we used big $O$ notation. After analyzing each step of the algorithm, it is concluded that the algorithm is $O(N \log N)$. However, it must be noted that this may not be the best way to determine the speed of the execution of the algorithm when running on a CPU.
Chapter 6

Conclusions and Future Work

6.1 Introduction

In this dissertation, we presented a modulation classification algorithm for cognitive radios that:

- is easy to extend for future modulation schemes,
- has fast processing time that would allow real-time MC, and
- is an optimization of previously presented methods.

6.2 Conclusions

Our contribution in this research includes three elements: a search for an efficient Gaussianity test in order to classify between single-carrier and multi-carrier modulations, a clustering algorithm to determine the type of constellation, and use of clustering algorithm for correcting I–Q and frequency imbalances and offsets in OFDM synchronization.

We have demonstrated that, among the well-known Gaussianity tests, the Cramer–von Mises test has the best overall performance. Along this process, we have also presented an alternative implementation to the Giannakis–Tsatsanis test that requires fewer computations than previous implementation proposed in Akmouche [1999] and Grimaldi et al. [2007]. This survey has included the nine
most-known Gaussianity tests and has tested them with different modulated signals for error rates and computation time. There was a void in the literature to find the most appropriate Gaussianity test to classify multi-carrier and single-carrier signals that we believe has been filled by this work.

We have also demonstrated that using a combination of $k$-means and $k$-center clustering on I–Q diagrams can be used as a classification method for single-carrier modulations that yields high efficiency. In order to demonstrate this and to come up with a way of classifying different modulations using this method, we have defined a parameter that we have named Cumulative Deviation Error or CDE. By using CDE, we were able to demonstrate that combining $k$-means and $k$-center algorithms significantly improves their performance compared to using any of them individually and we were able to actually measure this improvement. By applying these methods and by setting appropriate threshold levels, we achieved perfect classification rate for SNR > 5 dB when choosing between six different modulation schemes.

In Chapter 5, we have also compared the performance of this algorithm with a number of classifiers in the literature. It was shown that the performance of this algorithm is superior to the best published classifiers in being able to classify both QAM and PSK modulations with high accuracy in very low SNRs.

We have completed the MC algorithm by including OFDM feature extraction methods in the algorithm. We have demonstrated the performance of autocorrelation and cyclostationary tests in determining the existence of an OFDM signal and extracting its symbol duration and cyclic prefix duration. In this step, we have also proposed a method for correcting OFDM signal for frame time offset and I–Q phase imbalance by incorporating the clustering algorithm and using I–Q diagram of an OFDM signal. Our method shows very close linear dependency between the features extracted from the I–Q diagram and these offsets and imbalances. This can be used to correct these offsets and imbalances from OFDM signal. This method creates a great possibility in decreasing OFDM signal overhead or to design a new multi-carrier modulation scheme that won’t require as much overhead data for synchronization.

Finally, we have suggested a few options for implementing the MC algorithm on hardware and testing it with real on-air or off-air signals.
6.3 Future Research

For each of the contributions that we have made in this work, there is further work to be done. Some of them are listed below.

6.3.1 Automatic Constellation Detection

Currently, for single-carrier MC we are using a comparison between the results of clustering algorithms and a database of well-known modulation schemes. One of the drawbacks of this method is that it is not fully automated and, in order to expand it to new modulations, we need to include their I–Q diagram in the database.

However, a more logical method would be to devise a method independent of the modulation scheme. One idea for such a method would be to consider the distance between neighboring symbols. As shown in Figure 6.1, the number of adjacent symbols is only two in the case of PSK modulations but two, three, or four, depending on the position of symbol, in QAM modulations.

On the I–Q diagram, the symbols consist of two factors: the kernel (φ) and the distance metric. We can study them by determining which one provides the best scale to differentiate between modulation schemes. Choices for kernel functions include:

- Triangular
  \[
  \phi(x) = \begin{cases} 
  1 - |x|, & \text{if } |x| \leq 1 \\
  0, & \text{otherwise} 
  \end{cases}, \quad (6.1)
  \]

- Gaussian
  \[
  \phi(x) = e^{-x^2/2}, \quad (6.2)
  \]

- Exponential
  \[
  \phi(x) = e^{-|x|}, \quad (6.3)
  \]

Choices for distance metrics include:
Figure 6.1. Distance between symbols in I–Q diagram as a way of determining the type of modulation. (a) QPSK, (b) 8-PSK, (c) 16-QAM, and (d) a type of 8-QAM. Arrows indicate the neighboring symbols.

- Euclidean \((L - 2)\)

\[ d((x), (y)) = [(x_1 - y_1)^2 + (x_2 - y_2)^2]^{1/2}, \]  

\[ (6.4) \]

- \(L_p\)

\[ d((x), (y)) = [(x_1 - y_1)^p + (x_2 - y_2)^p]^{1/p}, \quad p > 0 , \]  

\[ (6.5) \]

- Hamming

\[ d((x), (y)) = |x_1 - y_1| + |x_2 - y_2| . \]  

\[ (6.6) \]
6.3.2 Optimization of Gaussianity Tests

Most of the available Gaussianity tests depend also on significance level, $\alpha$, and whether the distribution is upper-tailed or lower-tailed. For the significance level of these tests, we have used the default values in each test.

For further optimization it would be an interesting problem to study the performance of Gaussianity tests on real signals when $\alpha$ is one of variables of the simulations.

6.3.3 Protocol Analysis

Protocol analysis is the next phase in this research. Further parameter estimation and adding further complexities to the model is becoming the future step in this field of research. It includes analyzing protocol specifications and retrieve other protocol parameters than the modulation scheme, such as used frequency. For example, this would involve developing spectrum sensing techniques. It will also require analyzing higher layers after retrieving the parameters of the physical layer.

6.3.4 Hardware Implementation

Implementing this method on hardware and testing it using real-life signals can be a very important step in determining how well this algorithm can perform in real-life scenarios.

In the next step, we will implement this method on an available SDR board. Implementing the algorithm on hardware will require translating the codes in MATLAB to that of a system language such as Python or C that would allow the codes to be written on SDR boards. For a broader usage, we may also consider translating them to Verilog or VHDL languages that would allow the usage of the algorithms on any board equipped with an FPGA.

Some of the SDR available boards include

- Lyrtech SignalWAVe,
- Universal Software Radio Peripheral (USRP), and
- Lyrtech Small Form Factor (SFF) SDR Development Platform.
The first two boards are the boards we have worked with and we will continue to implement this algorithm on them.

### 6.3.4.1 Lyrtech SignalWAVe

Lyrtech SignalWAVe board was the first SDR board used to implement our algorithm. This board is equipped with a Xilinx Virtex-II FPGA that has 3 million gates and 32 Mb of DRAM. It also has a Texas Instruments’ TMS320C6713 DSP. For its input and output, it utilizes a 65 MS/s 14-bit ADC with Programmable Gain and a 125 MS/s 14-bit DAC. Its main advantage is that Simulink environment acts as a developer workbench for it, providing an easy way to do extensive testing before the final product.

Figure 5.2 shows a simple test by transmitting and receiving an OFDM signal using this board.

### 6.3.4.2 Universal Software Radio Peripheral

We have chosen Universal Software Radio Peripheral (USRP) for the hardware implementation of our algorithm due to its low-cost and the availability of open source, free software that is provided for integration with GNU radio.\(^1\) The USRP technical characteristics are as follows:

- an Altera Cyclone EP1C12Q240C8 FPGA
- four high-speed ADC, each capable of 64 MS/s at a resolution of 12 bits
- four high-speed DAC, each capable of 128 MS/s at a resolution of 14 bits
- a Cypress EZ–USB FX2 High-speed USB 2.0 controller
- 4 extension sockets (2 TX, 2 RX) in order to connect 2–4 daughterboards
- 64 GPIO pins available through 4 BasicTX/BasicRX daughterboards.

---

\(^1\)GNU Radio is a free software toolkit for learning about, building, and deploying software-defined radio systems. Started in 2001, GNU Radio is now an official GNU project [Blossom, 2010].
The tests for each algorithm will be performed according to the data flow in Figure 6.2. The USRP is used as a signal receiver to record three unique streams of data. These data are then input to the GNU Radio and passed through a modulator flow graph, which performs the required symbol mapping, modulation, up-conversion to RF, and addition of white Gaussian noise. The three RF signals are then processed to emulate the effects of being received on an $M$-element antenna array. The $M$ incident signals are passed through GNU Radio processing blocks to handle the signal reception, down-conversion, demodulation, and symbol extraction. Before the data is extracted in this receiver chain, the signal with carrier information is stripped off and sent to the MC algorithm. The results of each algorithm are then plotted using Python’s Matplotlib library.

As a first step, we have started with testing Gaussianity tests on real signals. The user interface that we have developed for this purpose can be seen in Figure 6.3. After choosing the desired test, the program returns 0 or 1 depending on whether the null hypothesis has been rejected or not.

### 6.3.5 Transmission Channels

Another step is to study this algorithm under different channel models. A realistic transmission channel model is an important step in evaluation of the performance of MC algorithms. The transmission channel can affect the transmitted signal either by introducing inter-symbol interference (ISI) or by lowering SNR level.

Throughout this work, we have studied the MC in AWGN channels. This has been the trend in previous works as well. Studying other channel models would be
Figure 6.3. User interface for choosing the desired Gaussianity test, written in MATLAB

an interesting topic of research.

While in Additive White Gaussian Noise (AWGN) channel a lower order QAM with a lower spectral efficiency is better than a higher order QAM with a better spectral efficiency in terms of $E_b/N_0$, in fading environments the opposite can be true. The explanation for this effect is that, in fading channels where a certain part of subcarriers can be totally lost in deep fades, the ability to correct for those lost subcarriers by having a large Hamming distance (as is the case for higher order QAM signals) is more important than having a large minimum Euclidean distance for each individual subcarrier (as is the case for lower order QAM signals) [Nee and Prasad, 2000]. For example, the rate 1/2 code combined with 16-QAM can tolerate more weak subcarriers than the rate 3/4 code with QPSK, resulting in an $E_b/N_0$ gain that is larger than the loss in Euclidean distance of 16-QAM versus QPSK.

In this section, we overview some of the more common transmission channel models.
6.3.5.1 Additive White Gaussian Noise (AWGN) Channels

AWGN channel is the most common transmission channel model. In this model the transmitted signal is degraded by thermal noise associated with the physical channel itself or other hardware used in the link. This model closely approximates cases such as space communications and forward path cable television.

6.3.5.2 Rayleigh Fading Channels

This model is often used in wireless mobile communications. In heavily built up urban environments, propagation can be best modeled using the Rayleigh fading model. When the received signal consists of a large number of plane waves, the received complex envelope $r(t) = r_I(t) + jr_Q(t)$ can be treated as a wide-sense stationary complex Gaussian random process.

If we assume 2–D isotropic scattering, $r_I(t)$ and $r_Q(t)$ are independent identically distributed (i.i.d.) zero-mean Gaussian random variables at any time $t_0$ with variance $b_0$. Under these conditions, the magnitude of the received complex envelope $\alpha(t) = |r(t)|$ has a Rayleigh distribution at any time $t_0$ [Stüber, 2001]:

$$p_\alpha(x) = \frac{x}{b_0} e^{-\frac{x^2}{2b_0}} . \tag{6.7}$$

The average envelope power is $E[\alpha^2] = \Omega_p = 2b_0$ so that

$$p_\alpha(x) = \frac{2x}{\Omega_p} e^{-\frac{x^2}{\Omega_p}} \quad x \geq 0 . \tag{6.8}$$

One of the popular models for simulating Rayleigh fading channel is Clarke’s model.

6.3.5.3 Rician Fading Channels

Some types of scattering environments have a specular or line of sight (LoS) component. In this case, $r_I(t)$ and $r_Q(t)$ are Gaussian random processes with non-zero means $m_I(t)$ and $m_Q(t)$, respectively. If we again assume that these processes are uncorrelated and the random variables $r_I(t)$ and $r_Q(t)$ have the same variance $b_0$, then the magnitude of the received complex envelope at time $t_0$ has a Rician
distribution [Stüber, 2001]:

\[ p_0(x) = \frac{x}{b_0} e^{-\frac{x^2}{2b_0}} I_0 \left( \frac{x s}{b_0} \right), \]  

(6.9)

where \( s^2 = m_f^2(t) + m_Q^2(t) \) is called the non-centrality parameter. This is defined as Rician fading and is very often observed in microcellular and mobile satellite applications.
Appendix A

Deriving Gaussianity using Kurtosis and Skewness of a Distribution

This appendix summarizes the background to Gaussianity tests in Chapter 3. It is driven from Stamatis [2003]

The theoretical normal distribution has a third standardized moment defined as

$$\gamma_1 = \sqrt{\frac{\mu_2^2}{\mu_2}}$$, \hspace{1cm} (A.1)

which provides a measure of the skewness of a distribution. For a normal distribution, $\gamma_1$ will be equal to zero; positive values indicate that the distribution has a greater tendency to tail to the right (positively skewed); negative values indicate a tailed distribution to the left (negatively skewed).

As with all other tests, the skewness analysis is based on a hypothesis test. Specifically, it assesses whether the sample data is so skewed that it is not probable that it was randomly drawn from a distribution that can be approximated by a normal model. Although a number of equations exist for this purpose, a common equation is as follows:

$$g_1 = \frac{\kappa_3}{\sigma^3}$$, \hspace{1cm} (A.2)

where $g_1$ is skewness index for sample data, $\sigma^3$ is the standard deviation of the
sample raised to the third power, and \( \kappa_3 \) is defined as

\[
\kappa_3 = \frac{n^2m_3 - 3nm_2 + 2m_1^3}{n(n-1)(n-2)},
\]  
(A.3)

where \( m_i \) is the \( i \)th sample moment about the mean.

The fourth standardized moment is a measure of the peakedness, or kurtosis, of the distribution. The normal distribution is said to be \textit{mesokurtic}, or intermediate, in nature. Distributions that are narrow with high peaks are said to be \textit{leptokurtic}. Flattened, broad distributions with high “shoulders” are said to be \textit{platykurtic}. The standardized fourth moment has the following theoretical form:

\[
\gamma_2 = \beta_2 = \frac{\mu_4}{\mu_2^2} - 3.
\]  
(A.4)

An important note should be raised at this point related to the kurtosis index. Many statisticians use equations derived from the equation above, which results in a calculated index of 0 for normal distributions. In other cases, however, equations are derived from the form:

\[
\gamma_2 = \frac{\mu_4}{\mu_2^2},
\]  
(A.5)

which results in a kurtosis index of 3 for the theoretical normal distribution. Neither approach is more correct, or more powerful, than the other. The only possible advantage of the first approach is that, while skewness is considered more desirable, it is essential that the basis for the formula used in a computer package or from a text reference is known in order to properly evaluate the results of the kurtosis analysis. A common calculational formula for kurtosis (where \( \text{E}(\gamma_2) = 0 \)) is

\[
g_2 = \kappa_4/\sigma^4,
\]  
(A.6)

where \( g_2 \) is kurtosis index for the sample data, \( \sigma^4 \) is standard deviation of the sample raised to the 4\(^{th} \) power, and \( \kappa_4 \) is defined as

\[
\kappa_4 = \frac{n^2(n+1)m_4 - 4(n+1)m_3m_1 - 3n(n-1)m_2^2 + 12nm_2m_1^2 - 6m_1^4}{n(n-1)(n-2)(n-3)}.
\]  
(A.7)

Testing the hypothesis of normality using the moment test is made relatively
simple, given that most computer statistical packages provide a measure of the skewness \((g_1)\) and kurtosis \((g_2)\) of the sample data.
Appendix B

The Giannakis–Tsatsanis Test: Derivation of Formulas

According to the original test [Giannakis and Tsatsanis, 1994], the theoretical fourth order cumulant $c_{4x}$ is consistently estimated by the sampled averages:

$$\hat{c}_{4x}(i_1, i_2, i_3) = \frac{1}{N} \sum_{i=0}^{N-i_1-1} x(i)x(i + i_1)x(i + i_2)x(i + i_3)$$

$$\quad - \left[ \frac{1}{N} \sum_{i=0}^{N-i_2-1} x(i)x(i + i_1) \right] \left[ \frac{1}{N} \sum_{i=0}^{N-i_2+i_3-1} x(i)x(i + i_2 - i_3) \right]$$

$$\quad - \left[ \frac{1}{N} \sum_{i=0}^{N-i_3-1} x(i)x(i + i_2) \right] \left[ \frac{1}{N} \sum_{i=0}^{N-i_3+i_1-1} x(i)x(i + i_3 - i_1) \right]$$

$$\quad - \left[ \frac{1}{N} \sum_{i=0}^{N-i_1-1} x(i)x(i + i_3) \right] \left[ \frac{1}{N} \sum_{i=0}^{N-i_1+i_2-1} x(i)x(i + i_1 - i_2) \right] \quad \text{(B.1)}$$

where $N$ denotes the number of samples.

According to the original test, all the cumulants $\hat{c}_{4x}(i_1, i_2, i_3)$, with $0 \leq i_1 \leq i_2 \leq i_3 \leq M - 1 < \infty$, of the received signal must be computed. Generally, we take $M \approx N^{0.4}$. All of these cumulants can be considered as components of a cumulant vector with a dimension $L = M(M + 1)(M + 2)/6$. But these components can be correlated, making it possible to reduce the number of calculations.

Computing $c_{4x}(0, t, t)$, with $0 \leq t \leq M < \infty$, can be done in the following
manners [Akmouche, 1999]

\[ \hat{c}_{4x}(0, t, t) = -\frac{1}{N} \sum_{i=0}^{N-t-1} x^4(i) + x^2(i)x^2(i + t), \quad (B.2) \]

or as a second version with fewer samples involved, we can also derive the following

\[ \hat{c}_{4x}(0, t, t) = -\frac{1}{N} \sum_{i=t}^{N-t-1} x^4(i) + x^2(i)x(i + t)x(i - t). \quad (B.3) \]

Based on our calculations, the most correct adaptation from Equation (B.1) would be Equation (B.3) and how it is derived is shown below,

\[ \hat{c}_{4x}(0, t, t) = \frac{1}{N} \sum_{i=0}^{N-1} x(i)x(i + t)^2 \]

\[- \left[ \frac{1}{N} \sum_{i=0}^{N-1} x^2(i) \right] \left[ \frac{1}{N} \sum_{i=0}^{N-1} x^2(i) \right] \]
\[- \left[ \frac{1}{N} \sum_{i=0}^{N-t-1} x(i)x(i + t) \right] \left[ \frac{1}{N} \sum_{i=0}^{N-t-1} x(i)x(i + t) \right] \]
\[- \left[ \frac{1}{N} \sum_{i=0}^{N-t-1} x(i)x(i + t) \right] \left[ \frac{1}{N} \sum_{i=0}^{N+t-1} x(i)x(i - t) \right] \]
\[\Rightarrow\]

\[ \hat{c}_{4x}(0, t, t) = \frac{1}{N} \sum_{i=0}^{N-1} x^2(i)x^2(i + t) \]

\[- \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x^2(i)x^2(j) \]
\[- \frac{1}{N^2} \sum_{i=0}^{N-t-1} \sum_{j=0}^{N-t-1} x(i)x(j)x(i + t)x(j + t) \]
\[\Rightarrow\]
\[\hat{c}_{4x}(0, t, t) = \frac{1}{N^2} \sum_{i=0}^{N-t-1} \sum_{j=0}^{N+t-1} x(i)x(j)x(i + t)x(j - t) \]
we can generally assume $x(i) \ll N$, so the equation can be approximated in the following manner

$$
\hat{c}_{4x}(0,t,t) \approx \frac{1}{N} \sum_{i=0}^{N-1} x^2(i)x^2(i+t) - x^4(i) - x^2(i)x^2(i+t) - x^2(i)x(i+t)x(i-t) .
$$

(B.6)

Also $x(i+t) = 0$ for $i > N - t$ and $x(i-t) = 0$ for $i < t$, which further results in

$$
\hat{c}_{4x}(0,t,t) \approx -\frac{1}{N} \sum_{i=t}^{N-t-1} x^4(i) + x^2(i)x(i+t)x(i-t) .
$$

(B.7)

The next step will be to compute the covariance matrix:

$$
\Sigma_c = \text{cov}\{c_{4x}(0,t_u,t_u), c_{4x}(0,t_v,t_v)\} .
$$

(B.8)

Finally, if the following inequality is verified

$$
d_{G,4} = c_{4x}^N \Sigma_c^{-1} c_{4x} < \tau_G ,
$$

(B.9)

where $\tau_G$ is an opportune set threshold, the Gaussianity is passed.

Vectors $c_r$ and $c_j$ have dimension $M$. Matrix $\sigma_{c_r}$ and $\sigma_{c_j}$ have size $M \times M$. Akmouche [1999] obtain a ratio $M(M+1)/3$ in reducing the size of vector and $M^2(M+1)^2/9$ in reducing the size of the matrix. Our method does not reduce the size of the matrix, but it slightly reduces the number of computations to calculate the vectors and the matrix.
Cluster-finding Algorithms in Pattern Recognition

Clustering problem in general is to split \( n \) points in \( R^d \) space into \( k \) groups. The aim is to minimize the intra-cluster similarity and maximize cross-cluster similarity. These algorithms are generally divided into five categories:

- Hierarchical
- Partitioning
- Density-based
- Grid-based
- Model-based

Throughout this appendix, we assume \( C(x) \) to be the closest cluster center to \( x \)

C.1 \( H \)-means

\( H \)-means is one of the simpler algorithms for the clustering problem [Spath, 1980]. This algorithm alternates between two procedures:

- using the given cluster centers, assign each point to the cluster with the nearest center;
• using the given cluster assignments, replace each cluster center by the centroid or average of the points in the cluster.

The problem with \( H \)-means algorithm is that this method could lead to empty clusters. As the centroid is recalculated and observations are reassigned to groups, some clusters could become empty.

C.2 \( k \)-means

This algorithm is summarized as follows:

For each point, move it to another cluster if that would lower the energy. If you move a point, immediately update the cluster centers of the two affected clusters.

The details for this algorithm, as part of our modulation classification algorithm, are provided in Chapter 4.

C.3 \( k \)-center

The criteria for this algorithm is given by

\[
\min \max_{x \in X} \| x - C(x) \|. \tag{C.1}
\]

The details for this algorithm are introduced in Chapter 4. We use this algorithm for initializing the \( k \)-means algorithm.

C.4 \( k \)-medians

\( k \)-medians is a variation of \( k \)-means clustering, where instead of calculating the mean for each cluster to determine its centroid, one instead calculates the median [Jain and Dubes, 1988]. The criteria for this algorithm is defined as

\[
\min \sum_{x \in X} \| x - C(x) \|. \tag{C.2}
\]
C.5 \textit{k–medians Squared}

This algorithm is very similar to \textit{k}\text{-medians} algorithm, but it is much more sensitive to outliers compared to other clustering algorithms. Its criteria are defined as

\[ \min \sum_{x \in X} \| x - C(x) \|^2. \] (C.3)

C.6 \textbf{Fuzzy c-means}

In fuzzy clustering, each point has a degree of belonging to clusters, as in fuzzy logic, rather than belonging completely to just one cluster [Bezdek, 1981]. Thus, points on the edge of a cluster may be in the cluster to a lesser degree than points in the center of cluster. The algorithm minimizes intra-cluster variance as well, but has the same problems as \textit{k}\text{-means}; the minimum is a local minimum, and the results depend on the initial choice of weights. The local minimum is important because it is very common for algorithms to get stuck at a solution that is merely a local minimum. For such a local minimum, every slight rearrangement of the solution makes the energy go up; however, a major rearrangement would result in a big drop in energy.

This algorithm has been tried on I–Q modulations for classification purposes in the literature [Mobasseri, 1999]. However, its complexity is one of its drawbacks compared to the method described in our work.

C.7 \textbf{QT Clustering}

QT (quality threshold) clustering is an alternative method of partitioning data, invented for gene clustering [Heyer et al., 1999].

The algorithm follows these steps:

\begin{itemize}
  \item Choose a maximum diameter for clusters.
  \item Build a candidate cluster for each point by iteratively including the point that is closest to the group, until the diameter of the cluster surpasses the threshold.
\end{itemize}
• Save the candidate cluster with the most points as the first true cluster, and remove all points in the cluster from further consideration.

• Recurse with the reduced set of points.

The distance between a point and a group of points is computed using complete linkage, i.e., the maximum distance from the point to any member of the group.

The QT algorithm requires more computing power than \( k \)-means, but does not require specifying the number of clusters \( a \ priori \), and always returns the same result when run several times. This algorithm can be a good alternative to our method, because it does not require the number of clusters \( a \ priori \). However, its great disadvantage is that it is computationally intensive. Increasing the minimum cluster size, decreasing the Minimum Correlation, or increasing the number of samples on the selected sample list can greatly increase the computational time.
Appendix D

Glossary

ALRT  Average Likelihood Ratio Test
ASK   Amplitude-shift Keying
AWGN  Additive White Gaussian Noise
CDE   Cumulative Deviation Error
CDF   Cumulative Distribution Function
CLT   Central Limit Theorem
CR    Cognitive Radio
FFT   Fast Fourier Transform
FPGA  Field-Programmable Gate Array
FSK   Frequency-shift Keying
GLRT  Generalized Likelihood Ratio Test
HLRT  Hybrid Likelihood Ratio Test
I–Q   Inphase–Quadrature
ICI   Inter-carrier Interference
IID   Independent Identically Distributed
ISI  Inter-symbol Interference
MC   Modulation Classification
MIMO Multiple Input–Multiple Output
ML   Maximum Likelihood
OFDM Orthogonal Frequency-Division Multiplexing
PDF  Probability Density Function
PSK  Phase-shift Keying
QAM  Quadrature Amplitude Modulation
SDR  Software-Defined Radio
SNR  Signal-to-Noise Ratio
USRP Universal Software Radio Peripheral
VHDL VHSIC Hardware Description Language
MATLAB Codes

This appendix includes some of our codes in MATLAB for Gaussianity tests and clustering algorithms.

E.1 Gaussianity Tests

E.1.1 Giannakis–Tsatsanis

```matlab
function [h, d_G] = GTtest(x)
    T = length(x);
    M = T^.4;
    sumelements = zeros(1,T);
    cum4_1 = zeros(1,ceil(M));
    t_G = 4.2*10^-4; % Threshold for 512 samples
    for eta = 1 : ceil(M)
        for t = eta+1 : T-eta
            sumelements(t) = -x(t).^2.*x(t+eta).*x(t-eta) - x(t).^4;
        end;
        cum4_1(eta) = 1/T*sum(sumelements);
    end;
    d_G = cum4_1 * cov(cum4_1)^-1 * cum4_1';
    if d_G <= t_G
        h = 0;
    else
        h = 1;
    end
    return,
```
E.1.2 Anderson–Darling

function [h,P] = ADtest(x,alpha)

switch nargin
    case{1}
        alpha = 0.05;
    case{2}
        if isempty(x) == false && isempty(alpha) == false
            if (alpha <= 0 || alpha >= 1)
                fprintf('Warning: Significance level error; must be 0 < alpha < 1 
');
                return;
            end
        end
    end
end

n = length(x);
if n < 10,
    disp('Sample size must be greater than 10.');
    return,
else
    x = x(:);
    x = sort(x);
    f_x = normcdf(x,mean(x),std(x));
    i = 1:n;
    S = sum(((2*i)-1)/n)*(log(f_x)+log(1-f_x(n+1-i)));
    AD = -n-S;
end
if (AD >= 0.00 && AD < 0.200);
    P = 1 - exp(-13.436 + 101.14*AD - 223.73*AD^2);
elseif (AD >= 0.200 && AD < 0.340);
    P = 1 - exp(-8.318 + 42.796*AD - 59.938*AD^2);
elseif (AD >= 0.340 && AD < 0.600);
    P = exp(0.9177 - 4.279*AD - 1.38*AD^2);
else (AD >= 0.600 && AD <= 13);
    P = exp(1.2937 - 5.709*AD + 0.0186*AD^2);
end
if P >= alpha;
    h = 0;
else
    h = 1;
end
return,

E.2 Gaussianity Test Simulation Code

% Simulating Normality tests - For calculation of error rate in Normality tests
% Written by Okhtay Azarmanesh
close all
clear all
warning off

%% initializing
DB_min = 3;
DB_max = 15;
n = 9; % Number of Normality tests
samp = 512; % Number of samples per signal
iteration = 2000; % Number of runs per simulation
err_all = zeros((DB_max-DB_min+1),n);
ti = zeros(iteration, n);
i = 1;
signif = 0.05;

%%
for db = DB_min:DB_max
    error=zeros(n,1);
    for j=1:iteration
        err = randn(samp,1)*10^(-db/10);
        X_1 = randn(samp,1);
        X_2 = randi(M,samp,1)-1;
        X_1 = modulate(h,X_2);
        % X = awgn(X_1,db);
        X = X_1 + err;
        [H,Threshold,t] = norm_test_time(real(X), signif);
        for count = 1:n
            if H(count)==1
                error(count)=error(count)+1;
            end;
        end;
        ti(j,:)=t;
        %assigning times from each calculation of different tests into a vector
    end;
    err_all(i,:)=error/iteration;
    %Average error for a correct normality detection per #iteration of simulations
    i=i+1;
end;

parfor i=1:(DB_max-DB_min+1)
    ti_ave(i,:) = mean(ti); %average time from 100 runs of norm test
end;

%% Plotting error vs. SNR
figure(1);
db = DB_min:DB_max;
semilogy(db, err_all(:,1),'-rx');
hold on
semilogy(db, err_all(:,2),'-bd');
semilogy(db, err_all(:,3),'-gp');
semilogy(db, err_all(:,4),'-c+');
semilogy(db, err_all(:,5),'-mo');
semilogy(db, err_all(:,6),':bs');
E.3 Clustering Algorithms

E.3.1 Main Code

close all;
clear all;
n = 1024; % Number of samples
SNR = 30; % after passing from an AWGN channel (in dB)
noise_factor = 10^(-SNR/20);

%%
% ***** For QAM modulations - SCALED to 1 watt ********

figure('Units','characters','Position',[30 5 135 50]);

for constel=1:4
    K = 2^(2*constel); % Create a signal source for QAM modulations.
    x = randintvec(n,K);
    h = modem.qammod('M',K,'PhaseOffset',0,'SymbolOrder','binary'); % Create a modulator object
    % scale = sqrt(modnorm(x,'avpow',1))
    y = modulate(h,x); % Modulate the signal x.
    y_chann = awgn(y ,SNR); % Output of AWGN channel

    subplot(2,2,2*constel); plot(real(y_chann), imag(y_chann),'.','MarkerSize',4)
    % I-Q diagram of output
    Data = [real(y_chann), imag(y_chann)];
    [~,u]=KCenterClustering(Data,K);
    [~, C1] = kmeans(Data, K, 'Start', u);
    hold on,
plot(C1(:,1),C1(:,2),'ro','MarkerEdgeColor','r','MarkerFaceColor','r',... 'MarkerSize',4);
end;
subplot(2,2,1); title('(a) 4-QAM');xlabel('I');ylabel('Q');
grid; axis('square'); axis([-2 2 -2 2])
subplot(2,2,2); title('(b) 16-QAM');xlabel('I');ylabel('Q');
grid; axis('square'); axis([-4 4 -4 4])
subplot(2,2,3); title('(c) 64-QAM');xlabel('I');ylabel('Q');
grid; axis('square'); axis([-10 10 -10 10])
subplot(2,2,4); title('(d) 256-QAM');xlabel('I');ylabel('Q');
grid; axis('square'); axis([-20 20 -20 20])

%%
% ***** For PSK modulations *****

figure('Units','characters','Position',[30 5 135 50]);

for constel=2:5
    K = 2^constel;
    x = randintvec(n,K); % Create a signal source for QAM modulations.
    h = modem.pskmod('M',K,'PhaseOffset',0,'SymbolOrder','binary'); % Create a modulator object
    y = modulate(h,x); % Modulate the signal x.
    y_chann = awgn(y,SNR); % Output of AWGN channel
    subplot(2,2,constel-1); plot(real(y_chann), imag(y_chann),'.','MarkerSize',4)
    Data = [real(y_chann), imag(y_chann)];
    [~,u]=KCenterClustering(Data,K);
    [~, C1] = kmeans(Data, K, 'Start', u);
    hold on,
    plot(C1(:,1),C1(:,2),'ro','MarkerEdgeColor','r','MarkerFaceColor','r',... 'MarkerSize',4);
end;
subplot(2,2,1); title('(a) QPSK');xlabel('I');ylabel('Q');
grid; axis('square'); axis([-2 2 -2 2])
subplot(2,2,2); title('(b) 8-PSK');xlabel('I');ylabel('Q');
grid; axis('square'); axis([-2 2 -2 2])
subplot(2,2,3); title('(c) 16-PSK');xlabel('I');ylabel('Q');
grid; axis('square'); axis([-2 2 -2 2])
subplot(2,2,4); title('(d) 32-PSK');xlabel('I');ylabel('Q');
grid; axis('square'); axis([-2 2 -2 2])

%%
% ***** For PAM modulations *****

figure('Units','characters','Position',[30 5 135 50]);
for constel=1:4
    K = 2^constel;
    x = randintvec(n,K);  % Create a signal source for QAM modulations.
    h = modem.pammod('M',K,'SymbolOrder','binary');  % Create a modulator object
    y = modulate(h,x);  % Modulate the signal x.
    y_chann = awgn(y ,SNR);  % Output of an AWGN channel
    subplot(2,2,constel); plot(real(y_chann), imag(y_chann),'.','MarkerSize',4)
    Data = [real(y_chann), imag(y_chann)];
    [~,u]=KCenterClustering(Data,K);
    [~, C1] = kmeans(Data, K, 'Start', u);
    hold on,
    plot(C1(:,1),C1(:,2),'ro','MarkerEdgeColor','r',
         'MarkerFaceColor','r','MarkerSize',4);
    end;
subplot(2,2,1); title('(a) 2-PAM');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])
subplot(2,2,2); title('(b) 4-PAM');xlabel('I');ylabel('Q');
grid; axis('square');axis([-4 4 -2 2])
subplot(2,2,3); title('(c) 8-PAM');xlabel('I');ylabel('Q');
grid; axis('square');axis([-10 10 -2 2])
subplot(2,2,4); title('(d) 16-PAM');xlabel('I');ylabel('Q');
grid; axis('square');axis([-20 20 -2 2])

%%
% ***** For MSK modulations *****
K = 2;
figure('Units','characters','Position',[30 5 130 50]);

x = randintvec(n,K);  % Create a signal source for QAM modulations.
h = modem.mskmod;  % Create a modulator object
y = modulate(h,x);  % Modulate the signal x.

y_chann = awgn(y ,SNR);  % Output of an AWGN channel

plot(real(y_chann), imag(y_chann),'.','MarkerSize',4)  % I-Q diagram of output
Data = [real(y_chann), imag(y_chann)];
[~,u]=KCenterClustering(Data,K);
[~, C1] = kmeans(Data, K, 'Start', u);
hold on,
plot(C1(:,1),C1(:,2),'ro','MarkerEdgeColor','r',
     'MarkerFaceColor','r','MarkerSize',4);
title('MSK'); xlabel('I'); ylabel('Q'); axis([-2 2 -2 2])
grid; axis('square');

%%
% ***** For DPSK modulations *****
for constel=1:4
    K = 2^constel;
x = randintvec(n,K);  \% Create a signal source for QAM modulations.
h = modem.dpskmod('M',K,'PhaseRotation',0,'SymbolOrder','binary');
    \% Create a modulator object
y = modulate(h,x);  \% Modulate the signal x.
y_chann = awgn(y ,SNR);  \% Output of AWGN channel
end;

subplot(2,2,1); title('(a) 2-DPSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])

subplot(2,2,2); title('(b) 4-DPSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])

subplot(2,2,3); title('(c) 8-DPSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])

subplot(2,2,4); title('(d) 16-DPSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])

\%
\% ***** For FSK modulations *****

freqsep = 32;
namp = 8;
Fs = 512;

figure('Units','characters','Position',[30 5 130 50]);

for constel=1:4
    K = 2^constel;
x = randi(K,n,1)-1;  \% Create a signal source for FSK modulations.
y = fskmod(x, K, freqsep, namp, Fs);  \% Create a modulator object
y_chann = awgn(y ,SNR);  \% Output of AWGN channel
end;

subplot(2,2,1); title('(a) 2-DSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])

subplot(2,2,2); title('(b) 4-DSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])

subplot(2,2,3); title('(c) 8-DSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])

subplot(2,2,4); title('(d) 16-DSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])
'MarkerFaceColor','r',...
'MarkerSize',4);
end;
subplot(2,2,1); title('(a) 2-FSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])
subplot(2,2,2); title('(b) 4-FSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])
subplot(2,2,3); title('(c) 8-FSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])
subplot(2,2,4); title('(d) 16-FSK');xlabel('I');ylabel('Q');
grid; axis('square');axis([-2 2 -2 2])

E.3.2 $k$-center Function

function [cluster,h]=KCenterClustering(Data,K)

% randomly select an object as the initial point
Init = cell(rand(1)*size(Data,1));
h = zeros(K,size(Data,2));
h(1,:) = Data(Init,:);

% for j = 1:size(Data,1)
% dist(j) = sum( (Data(j,:)-h(1,:)).^2 );
% cluster(j) = 1;
% end

dist = sum( (Data-repmat(h(1,:),[size(Data,1),1])).^2,2 )
cluster = ones(1,size(Data,1));
dist1=dist;
Hind=Init;
for i = 2:K
    dist1(Hind)=-1;
    D = max(dist1);
    ind = find(dist==D);
    Hind=[Hind,ind(1)];
h(i,:) = Data(ind(1),:);
    % for j = 1:n
    %     if sum( (Data(j,:)-h(i,:)).^2 ) < dist(j)
    %         dist(j) = sum( (Data(j,:)-h(i,:)).^2 );
    %         dist1(j) = sum( (Data(j,:)-h(i,:)).^2 );
    %         cluster(j)=i;
    %     end
    % end
    distnew = sum( (Data-repmat(h(i,:),[size(Data,1),1])).^2,2 )
    cluster(find(distnew-dist<0))=i;
    dist(find(distnew-dist<0))=distnew(find(distnew-dist<0));
    dist1=dist;
end
E.4 OFDM Simulations

E.4.1 Clustering Algorithms for OFDM Signals - Main Code

% Written by Okhtay Azarmesh,
% February 11, 2011
% Part of the research on Modulation Classification

% Carrier Frequency Offset range = [-1, 1]
% Frame Time Offset range = [-1, 1]
% Sampling Clock Offset range = [-1, 1]
% I-Q Gain Imbalance range = [-0.2, 0.2]
% I-Q Phase Imbalance range = [-0.2, 0.2]
% I-Q Differential time range = [-0.2, 0.2]
% 
% clear all;
close all;
offset = 0.1; \%Percentage of Offset/100
imbalance = 0.3; \%Percentage of Imbalance/100
figure('Units','characters','Position',[30 5 130 50]);

%Without any offset
D_offset = [0,0,0,0,0,0];
figure(1); subplot(2,2,1)
OFDM_offset_zeroremoval(D_offset);
title('(a) OFDM Signal')
xlabel('I'); ylabel('Q');

%Carrier Frequency Offset
D_offset = [offset,0,0,0,0,0];
figure(1); subplot(2,2,2)
OFDM_offset_zeroremoval(D_offset);

% Frame Time Offset
D_offset = [0,offset,0,0,0,0];
figure(1); subplot(2,2,3)
OFDM_offset_zeroremoval(D_offset);

% Sampling Clock Offset
D_offset = [0,0,offset,0,0,0];
figure(1); subplot(2,2,4)
OFDM_offset_zeroremoval(D_offset);
% Initializing
d_frq = D_parameter(1); %Carrier Frequency Offset
d_tm = D_parameter(2); %Frame Time Offset
d_clk = D_parameter(3); %Sampling Clock Offset
d_gn = D_parameter(4); %I-Q Gain Imbalance
d_phs = D_parameter(5); %I-Q Phase Imbalance
del_t = D_parameter(6); %I-Q Differential time

hh1 = sinc(-4+del_t:4+del_t).*kaiser(9,5);
hh1 = hh1/sum(hh1);
hh2 = [0 0 0 0 1 0 0 0 0];
hh1a = sinc(-4+d_tm:4+d_tm).*kaiser(9,5);’;
hh1a = hh1a/sum(hh1a); 
hh1b = sinc(-4-0.1:4-0.1).*kaiser(9,5);’;
hh1c = sinc(-4+0.1:4+0.1).*kaiser(9,5);’;
hh1d = 5*(hh1b-hh1c);

for mm=1:30
% generate reference data
fd1 = zeros(1,128);
dx1 = (floor(4*rand(1,52))-1.5)/1.5;
dy1 = (floor(4*rand(1,52))-1.5)/1.5;
dx1(27) = 0;
dy1(27) = 0;
%go to time domain
fd1(64-26:64+25) = dx1+1i*dy1;
fd1 = fftshift(fd1);
d1 = ifft(fd1);

% delay quadrature response relative to in-phase response
x1=real(d1);
y1=imag(d1);
%add cyclic prefix and then delay quad signal and strip prefix
x2=[x1(120:128) x1 x1(1:9)];
y2=[y1(120:128) y1 y1(1:9)];

x3=conv(x2,hh2);
y3=conv(y2,hh1);

x4=x3(14:141);
y4=y3(14:141);
% differential delay on I and Q has been inserted

% insert gain and phase imbalance
dia=x4+1i*(1+d_gn)*y4*exp(1i*d_phs);
% gain and phase imbalance inserted

%adding frequency offset
dia=dia.*exp(-1i*2*pi*(-64:63)*d_frq/128);
%shifted spectrum

%now insert block delay
x1=real(dia);
y1=imag(dia);
%add cyclic prefix and then delay quad signal and strip prefix
x2=[x1(120:128) x1 x1(1:9)];
y2=[y1(120:128) y1 y1(1:9)];
x3=conv(x2,hh1a);
y3=conv(y2,hh1a);
x4=x3(14:141);
y4=y3(14:141);
dia=x4+i*y4;

% block delay on I and Q has been inserted

% now insert clock offset
xi=real(dia);
yi=imag(dia);

% add cyclic prefix and then delay quad signal and strip prefix
x2=[x1(120:128) x1 x1(1:9)];
y2=[y1(120:128) y1 y1(1:9)];

x3=conv(x2,hh1d);
y3=conv(y2,hh1d)
x3a=conv(x2,hh2);
y3a=conv(y2,hh2);

x3b=x3a+(d_clk/153)*(-77:76).*x3;
y3b=y3a+(d_clk/153)*(-77:76).*y3;

x4=x3b(14:141);
y4=y3b(14:141);
dia=x4+i*y4;

% block delay on I and Q has been inserted

% clock offset inserted
fd2a=fftshift(fft(dia));
fdet2(mm,:)=fd2a(64-(-28:28));
f_avg(mm,:)=fd2a;
end
plot(fdet2(:,1:57),'.b','MarkerSize',5)

%========== Clustering ===========
const_size = 16; %Constellation size
dimension = size(fdet2);

% convert the dimension of fdet2 from 16x114 to 1710x1
farray = reshape(fdet2, 1, dimension(1)*dimension(2));
Data = [real(farray'), imag(farray')];
[~,u]=KCenterClustering(Data,const_size);

hold on;
plot(C1(:,1),C1(:,2),'rp','MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',8);
grid
axis('square')
E.5 BPSK vs. QPSK Classifier for Comparing with Other Classifiers

E.5.1 Main Code

clear all
clc
RUN = 1000;
ratio = RUN/100;
Class_4PSK = zeros(1,2);
SNR = 10;
tic
for run = 1:RUN
    outcome = Classify_comparison_with_Dobre(SNR);
    Class_4PSK = Class_4PSK+outcome;
    progress_percent = int16(run/ratio);
    progress(progress_percent)
end;
toc
Class_4PSK/ratio

E.5.2 Classifier Function

function outcome = Classify_comparison_with_Dobre(SNR)

K = 4;
outcome = zeros(1,2);
n = 100;   % Number of samples

%% QPSK
for i = 1:4
    C_q(i) = exp(2*1i*pi*(i-1)/4);
end;
C_4PSK = [real(C_q);imag(C_q)]';
constel_pow = sum(sum(C_4PSK.^2,2));

%% BPSK
for i = 1:2
    C_b(i)=exp(2*1i*pi*(i-1)/2);
end;
C_2PSK_r = [real(C_b);imag(C_b)]';
C_2PSK = [C_2PSK_r;C_2PSK_r];
%
[~, C1] = FindCCenterPSK(SNR,n,K);
Cluster_error(1) = sum(sum(C_diff_1.^2,2))/constel_pow;
Cluster_error(2) = sum(sum(C_diff_2.^2,2))/constel_pow;

%% Classification
Threshold = 0.55;
for i = 1:2
    if Cluster_error(i)< Threshold
        outcome(i) = 1;
    end
end
Bibliography


Bowman, K., and L. Shenton (1975), Omnibus test contours for departures from normality based on $\sqrt{b_1}$ and $b_2$, Biometrika, 62, pp. 243–250.


D’Agostino, R. B., and M. A. Stephens (Eds.) (1986), Goodness-of-Fit Techniques, Marcel Dekker, Inc.


Fisher, R. A. (1928), On a property connecting the $\chi^2$ measure of discrepancy with the method of maximum likelihood, Atti del Congresso Internazionale dei Matematici, Bologna, 6, pp. 95–100.


IEEE 1900.1 (2008), http://grouper.ieee.org/groups/scc41/1/.


Vita

Okhtay Azarmanesh

OKHTAY AZARMANESH received his B.Sc. in Electrical Engineering from Sharif University of Technology and his M.Sc. in Aerospace Communication Systems from Télécom Paris (École Nationale Supérieure des Télécommunications, now Télécom ParisTech) and SUPAERO (École Nationale Supérieure de l’Aéronautique et de l’Espace). His research interests include software-defined radios, cognitive radios, wireless communications, and satellite communications.