

**AN INFORMATION-BASED COMPLEXITY APPROACH TO
ACOUSTIC LINEAR STOCHASTIC TIME-VARIANT SYSTEMS**

by

Juan Bautista Valera-Márquez

A dissertation submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTING AND INFORMATION SCIENCES AND ENGINEERING

University of Puerto Rico

Mayagüez Campus

May, 2013

Approved by:

Vidya Manian, Ph.D.
Member, Graduate Committee

Date

Lizdabel Morales, Ph.D.
Member, Graduate Committee

Date

Manuel Rodríguez, Ph.D.
Member, Graduate Committee

Date

Kejie Lu, Ph.D.
Member, Graduate Committee

Date

Domingo Rodríguez, Ph.D.
President, Graduate Committee

Date

Esov Velázquez, Ph.D.
Representative of Graduate Studies

Date

Wilson Rivera, Ph.D.
CISE Ph.D. Program Coordinator

Date

Abstract of Dissertation Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of DOCTOR OF PHILOSOPHY

**AN INFORMATION-BASED COMPLEXITY APPROACH TO
ACOUSTIC LINEAR STOCHASTIC TIME-VARIANT SYSTEMS**

By

Juan Bautista Valera-Márquez

May 2013

Chair: Domingo Rodríguez

Major Department: Electrical & Computer Engineering

This thesis describes the formulation of a *Computational Signal Processing* (CSP) modeling framework for the analysis of *underwater acoustic signals* used in the *search, detection, estimation, and tracking* (SDET) operations of moving objects. The underwater acoustic medium where the signals propagate is treated as linear stochastic time-varying system exhibiting double dispersive characteristics, in time and frequency, simultaneously.

Acoustic Linear Stochastic (ALS) time-variant systems are characterized utilizing what is known as *time-frequency calculus*. The interaction of wavefront acoustic pressure fields with underwater moving objects is modeled using what is termed *Imaging Sonar and Scattering* (ISS) operators. It is demonstrated how the proposed CSP modeling framework, called ALSISS, may be formulated as an aggregate of ALS systems and ISS operators. Furthermore, it is demonstrated how concepts, tools, methods, and rules from the field of *Information-Based Complexity* (IBC) are utilized to seek approximate solutions to **NP**-hard problems encountered in the analysis of underwater acoustic signals treated under the ALSISS modeling framework.

Error approximation algorithms, formulated as approximate solutions, are implemented using *convex optimization* techniques. Finally, *Kronecker products algebra* is used as a mathematical language to formulate new variants of *matching pursuit* algorithms and to aid in the mapping of these algorithms to parallel computational structures.

Resumen de Disertación Presentado a la Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de DOCTOR EN FILOSOFÍA

**UN ENFOQUE DE COMPLEJIDAD BASADA EN INFORMACIÓN
PARA SISTEMAS LINEALES ACÚSTICOS ALEATORIOS
VARIANTES EN EL TIEMPO**

Por

Juan Bautista Valera-Márquez

Mayo 2013

Consejero: Domingo Rodríguez
Departamento: Ingeniería Eléctrica y Computadoras

Este trabajo de tesis describe la formulación de un arquetipo de modelado *computacional para el procesamiento de señales* (CSP, por sus siglas en inglés), con el fin de analizar *señales acústicas submarinas* usadas en operaciones de *busqueda, detección, estimación y rastreo* (SDET, por sus siglas en inglés) de objetos móviles. El medio acústico submarino donde las señales se propagan es tratado como un sistema lineal, aleatorio, variante en el tiempo, que exhibe características de doble dispersión, en tiempo y en frecuencia, de manera simultánea.

Los *sistemas variantes en el tiempo, acústicos, lineales, aleatorios* (ALS, por sus siglas en Inglés) son caracterizados utilizando lo que se conoce como *cálculo tiempo-frecuencia*. La interacción de un campo acústico de presión de ondas con objetos que se mueven bajo el agua es modelada usando operadores de *sonar de imágenes y dispersión* (ISS, por sus siglas en inglés). En este trabajo de tesis se demuestra cómo el arquetipo CSP propuesto para el modelado, llamado ALSISS, es formulado como una composición o agregado de sistemas ALS y operadores ISS. Además, se

demuestra cómo conceptos, herramientas, métodos y reglas del campo de la *complejidad basada en información* (IBC, por sus siglas en Inglés) son utilizados para buscar soluciones aproximadas a problemas **NP**-hard encontrados en el análisis de señales acústicas submarinas tratadas bajo el arquetipo de modelado ALSISS. Los algoritmos de aproximación de error, formulados como soluciones aproximadas, son implementados usando técnicas de *optimización convexa*. Finalmente, un *álgebra de productos Kronecker* es usada como lenguaje matemático para formular nuevas variantes de *algoritmos de búsqueda de coincidencias* (*matching pursuit* por su nombre en inglés) y para ayudar en la conversión de dichos algoritmos en estructuras computacionales paralelas.

Copyright © 2013

by

Juan Bautista Valera-Márquez

To God, my Lord and my Savior.

Acknowledgments

I deeply appreciate the meaningful contribution to my academic formation given by *Dr. Domingo Rodriguez* who has been my mentor, my professor, and my friend.

I would like to thank my advisory committee: *Dr. Manuel Rodríguez*, *Dr. Kejie Lu*, *Dr. Vydia Manian*, and *Dr. Lizabeth Morales*, for their encouragement and support.

I feel a debt of gratitude for the administrative support provided by Ms. Sandra Montalvo (Graduate Academic Counselor) and Dr. Pedro I. Rivera-Vega (ECE Department Head).

I have a special gratitude to my friends *Siegwart Mayr*, *Gabriel Larrius*, and *Brett Richards* who worked on the grammar checking of this thesis.

I would like to acknowledge the collaboration provided by the master's degree students, *Juan Pablo Soto-Quirós*, *David Márquez*, and *Ángel Camelo*, who were part of the research team at the Automatic Information Processing (AIP) Laboratory (Research and Develop Center, UPRM).

This work was supported in part by National Science Foundation (NSF) under grants CNS-0922996 and CNS-0424546.

Table of Contents

Abstract English	ii
Abstract Spanish	iv
Acknowledgments	viii
List of Tables	xii
List of Figures	xiv
List of Abbreviations	xviii
List of Symbols	xx
1 Introduction	1
1.1 Justification & Problem Formulation	2
1.2 Problem Space	6
1.2.1 Continuous Ambiguity Functions and the General Class of Cohen Distributions	9
1.2.2 Discrete Ambiguity Functions (DAF) and Discrete Cohen Distributions (DCD)	10
1.2.3 Relating DAF and DCD Operations	11
1.3 Proposed Solutions	12
1.4 Thesis Organization	13
2 Elementary Concepts and Ideas	14
2.1 Introduction	14
2.2 Kronecker Products	14
2.2.1 Kronecker Products and Parallel Formulations	15
2.2.2 Kronecker Products and Vector Formulations	16
2.3 Modeling and Simulation	18
2.4 The Stochastic Processes	19
2.5 Random Fields	20

3	Background and Related Works	23
3.1	Literature Review	23
3.1.1	Introduction	23
3.1.2	Characterization of Randomly Time-Variant Linear Channels	24
3.1.3	Information-Based Complexity	27
3.1.4	Improved RIP Analysis of Orthogonal Matching Pursuit	30
4	IBC & CSP Theoretical Frameworks	32
4.1	Introduction	32
4.2	Foundations of Computing	32
4.2.1	System or Machine Abstraction	33
4.2.2	Complexity Classes	35
4.2.3	Finite-State Automaton	36
4.2.4	Formal Definition of a Turing Machine	38
4.3	Classical Complexity Measures	40
4.4	Information-Based Complexity (IBC) Definition	42
4.5	Assumptions of IBC	44
4.6	Application Fields for IBC	48
4.7	Works on IBC Field	50
4.8	Computational Signal Processing Framework	51
4.9	Convex Optimization	55
4.9.1	General Optimization Problem	55
4.9.2	Particular Optimization Problems	56
4.9.3	Linear Programming Problem	56
4.9.4	Convex Optimization Problem	56
4.9.5	Least-Square Problem	56
4.9.6	Approximation Algorithms	57
5	Acoustic Linear Stochastic Systems	60
5.1	Introduction	60
5.2	Linear Systems	60
5.3	Stochastic Systems	61
5.4	Time-Variant Systems	62
5.5	Study of Communication Systems	63
5.6	Estimation Using a Parallel Approach	70
5.6.1	The Channel as an Operator	70
5.6.2	Characterization Function for an ALS channel	71
5.6.3	Coherence Time T_C and Coherence Bandwidth B_C in the Channel Modeling	73
5.7	Implementation Results	74

6	MIMO Channel Parameter Estimation Algorithms Implementation	78
6.1	Introduction	78
6.2	Channel Configurations	78
6.2.1	SISO Case	79
6.2.2	MISO Case	80
6.2.3	SIMO Case	82
6.2.4	MIMO Case	84
6.3	Delay-Doppler Estimation Approaches	87
6.3.1	Delay-Doppler SISO Approach	87
6.3.2	Estimating a Delay-Doppler SISO ALS Channel	89
6.3.3	Delay-Doppler SISO Estimation Strategy using Matching Pursuit Algorithms	92
6.3.4	Matching Pursuit Complexity	94
6.3.5	Delay-Doppler SISO Channel Estimation Results	96
6.3.6	Delay-Doppler MIMO Approach	99
6.3.7	Estimating a Delay-Doppler MIMO ALS channel	101
6.3.8	Delay-Doppler MIMO Estimation Strategy	106
6.3.9	Delay-Doppler MIMO Channel Estimation Results	109
6.4	Parallel Modeling Tools	120
6.4.1	pMatlab	120
6.5	MIMO Channel Estimation Parallel Approach	122
6.6	MIMO ALS Parallel Approach	128
6.6.1	Computational Formulation	129
6.6.2	Kuck's Diagrams Representation	130
6.6.3	Data Structures	132
6.7	Testbeds	137
7	Ethical Considerations	144
7.1	Introduction	144
7.2	Sea and Ecosystems	144
7.3	Environment Issues	145
7.4	Legal Issues	147
7.5	Other Considerations	147
8	Conclusions and Future Works	148
8.1	Conclusions	148
8.2	Future Works	149

List of Tables

4-1	“Big O ” Notation Associated to Classical Complexity Class	41
4-2	Matlab Computation for Two Approaches (Discrete and Real-Number)	49
4-3	Comparative Table of Results of H_k and H_f Transforms	50
5-1	Bello’s Kernel Functions	63
6-1	Complexity of Matching Pursuit (MP) Algorithm Variants. K = Windows Length, D = Number of Delays, L = the Doppler Shifts, M = Number of Transmitters, N = Number of Receivers, and I = Algorithm Iterations.	95
6-2	Sequence of selected columns c_i on MP variants. Matrix \mathbf{X} has 150 columns.	95
6-3	Estimation of Delay-Doppler Spread Function using Matching Pursuit	97
6-4	2×2 MIMO Case. Estimation of delay-Doppler Spread Function using Matching Pursuit Greedy Algorithms. $\mathbf{U}_{0,0}$ function. Positions 0- 149 in \mathbf{U}_{MIMO}	110
6-5	2×2 MIMO Case. Estimation of delay-Doppler Spread Function using Matching Pursuit Greedy Algorithms. $\mathbf{U}_{1,0}$ function. Positions 150- 299 in \mathbf{U}_{MIMO}	111
6-6	2×2 MIMO Case. Estimation of delay-Doppler Spread Function using Matching Pursuit Greedy Algorithms. $\mathbf{U}_{0,1}$ function. Positions 300- 449 in \mathbf{U}_{MIMO}	112
6-7	2×2 MIMO Case. Estimation of delay-Doppler Spread Function using Matching Pursuit Greedy Algorithms. $\mathbf{U}_{1,1}$ function. Positions 450- 599 in \mathbf{U}_{MIMO}	113
6-8	Sequence of chosen columns c_i in Matching Pursuit algorithm variants. Using a \mathbf{X}_{MIMO} matrix of 600 columns. Part 1. Iterations 1-40. . .	114
6-9	Sequence of chosen columns c_i in Matching Pursuit algorithm variants. Using a \mathbf{X}_{MIMO} matrix of 600 columns. Part 2. Iterations 41-69. . .	115

6–10 Script to Create a 2-D Mapping	121
6–11 Script to Create a 3-D Mapping	122
6–12 Matching Pursuit Algorithm	123
6–13 Max_Proj Matlab Function	124
6–14 pMatlab Distributed Map Definition	126
6–15 Matlab/pMatlab Code for Distributed Column Selection	126
6–16 Script to Create a 2-D Mapping for Distributing the Matrix Z	133
6–17 Script to Create a 2-D Mapping for Distributing Channel Parameters.	135
6–18 Initialization of Local Variables.	136

List of Figures

1-1	Satellite view of Lake Maracaibo. Source: ICLAM	3
1-2	Graphical Representation of the Bathymetry of Lake Maracaibo system. Source: ICLAM	4
1-3	Oil Rigs in Lake Maracaibo	5
1-4	Hundreds of derricks pump oil from Lake Maracaibo.	5
2-1	Stochastic Process \mathcal{X} with its Stochastic Variables X_k	21
2-2	George Christakos showed in his work “ <i>Random Fields Models In Earth Sciences</i> ” which are the Mathematics Behind the Random Fields Theory	22
3-1	Relationship Between System Functions in Time-Variant Channels . . .	26
3-2	IBC Framework Spaces	28
4-1	Number of operations in n^4 , 2^{2n} , and $n!$ complexities.	36
4-2	Graphical Representation of a Finite State Automaton	37
4-3	Non-Deterministic Computation Tree	40
4-4	Complexity Classes: Polynomial (P), Non-Deterministic Polynomial (NP), NP-Complete, and NP-Hard	41
4-5	Traub-Werschulz IBC Framework	43
4-6	IBC Trade-Off	44
4-7	Complete Transformation Process using Two Different Approaches . . .	45
4-8	IBC Spaces. Ambiguity Function Case	46
4-9	Four Worlds Described by IBC Theory	46
4-10	A Reformulation of the Four Worlds Described in IBC Theory	47

4-11	Modified Traub-Werschulz IBC Framework	47
4-12	Information Space to Solution Space Transformation	51
4-13	Information Space to Evaluation Space Transformation	52
4-14	Computational Signal Processing (CSP) Framework	52
4-15	Evaluation Space to Solution Space Transformation	54
4-16	Approximation Algorithm Approach	58
5-1	Time-Frequency System Functions for LTV Channels	67
5-2	Relationship Between Delay-Doppler Spread Function and the Kernels $K_1(t, s)$ and $K_2(f, l)$	68
5-3	Relationship Between the Ambiguity Function, the Wigner Distribu- tion, and the Correlation Function	69
5-4	Enhanced Diagram Depicting Relationship Between the Ambiguity Func- tion, the Wigner Distribution, the Correlation Function, and the Delay-Doppler Spread Function $U(\nu, \xi)$	70
5-5	Functions $h(t)$ vs. $g(t, \xi)$: (a) LTI Systems (b) ALS Systems	73
5-6	ISS System: Two Nearby Scatterers using Square Pulses.	75
5-7	ISS System: Two Nearby Scatterers using Square Pulses.	75
5-8	ISS System: Two Nearby Scatterers using Sinusoidal Pulses.	75
5-9	ISS System: Two Nearby Scatterers using Sinusoidal Pulses.	75
5-10	ISS System: Two Nearby Scatterers using Optimized Chirp Pulses (MMPP).	76
5-11	ISS System: Two Nearby Scatterers using Optimized Chirp Pulses (MMPP).	76
5-12	ISS System: Two Nearby Scatterers using Square Pulses.	76
5-13	ISS System: Two Nearby Scatterers using Square Pulses.	76
5-14	ISS System: Two Nearby Scatterers using Sinusoidal Pulses.	77
5-15	ISS System: Two Nearby Scatterers using Sinusoidal Pulses.	77
5-16	ISS System: Two Nearby Scatterers using Optimized Chirp Pulses (MMPP).	77

5-17 ISS System: Two Nearby Scatterers using Optimized Chirp Pulses (MMPP).	77
6-1 Schematic of SISO Channel System	79
6-2 Schematic of MISO Channel System	81
6-3 Schematic of SIMO Channel System	82
6-4 Schematic of MIMO Channel System	84
6-5 ALS Channel under SISO Assumption	88
6-6 Delay-Doppler Spread Function estimated using Matching Pursuit Algorithms: (a) U Given (b) U Est. via ORLSMP (c) U Est. via OMP (d) U Est. via BMP	99
6-7 ALS Channel under the MIMO assumption	101
6-8 2×2 MIMO case. Delay-Doppler Spread Function $U_{0,0}$ estimated using Matching Pursuit Algorithms	116
6-9 2×2 MIMO case. Delay-Doppler Spread Function $U_{1,0}$ estimated using Matching Pursuit Algorithms	117
6-10 2×2 MIMO case. Delay-Doppler Spread Function $U_{0,1}$ estimated using Matching Pursuit Algorithms	118
6-11 2×2 MIMO case. Delay-Doppler Spread Function $U_{1,1}$ estimated using Matching Pursuit Algorithms	119
6-12 How <i>dmat</i> datatype distribute a bi-dimensional array between 4 processors	120
6-13 How <i>dmat</i> datatype distribute a 3-dimensional array between 8 processors	122
6-14 Single Program Multiple Data Paradigm Applied to Selection Column Procedure in OMP and BMP Algorithms	125
6-15 Timeline in Serial and Parallel Approaches. Case: 16 Columns and 4 Processors	127
6-16 MIMO ALS Channel Structure	129
6-17 Kuck's Diagram to Parallel MIMO ALS Estimation Problem.	132
6-18 MIMO ALS Input Matrix Z (Size $V \times S$) Containing each Input Signal $z_s[v]$	134
6-19 Data partition on 3D Attenuation Matrix	137

6-20	Ambiguity Function Surface $A_z \in l^2(\mathbb{Z}_{2^{16}})$ in a 3D Representation. Six Square Pulses.	138
6-21	Ambiguity Function Surface $A_z \in l^2(\mathbb{Z}_{2^{16}})$ in a 2D Representation. Six Square Pulses.	139
6-22	Ambiguity Function Surface $A_z \in l^2(\mathbb{Z}_{2^{16}})$. Chirp Pulse.	139
6-23	Ambiguity Function Surface $A_z \in l^2(\mathbb{Z}_{2^{16}})$ in a 2D Representation. Chirp Pulse.	140
6-24	Computer-Based ALS MIMO Channel Testbed. WS-1 and WS-2 are Dell Precision T7500 Workstations with Dual-Quad Core and 48GB RAM	141
6-25	Computer-Based ISS Testbed. WS-1 and WS-2 are Dell Precision T7500 Workstations with Dual-Quad Core and 48GB RAM	142
6-26	Integrated ALS-ISS System	143
7-1	Underwater Ecosystem Impacted by ALS Channels Research	145

List of Abbreviations

AF	Ambiguity Function.
AR	Autoregressive.
AoA	Angle of Arrival.
BCJR	Bahl-Cocke-Jelinek-Raviv.
BMP	Basic Matching Pursuit.
CAF	Cross Ambiguity Function.
CONFAC	Constrained Factor.
CSP	Computational Signal Processing.
DAF	Discrete Ambiguity Function.
DCDs	Discrete Cohen Distributions.
DET	Discrete Evolutionary Transform.
DPD	Direct Position Determination.
DTM	Deterministic Turing Machine.
DWD	Discrete Winer Distribution.
FDTE	Frequency Domain Turbo Equalizer.
FFT	Fast Fourier Transform.
FSA	Finite State Automaton.
GC	General Class.
IBC	Information-Based Complexity.
ICI	Inter-Channel Interference.
ISI	Inter-Symbol Interference.
LFM	Linear Frequency Modulation.
LLRs	Log-Likelihood Ratios.
LTI	Linear Time Invariant.
LTV	Linear Time Variant.
MCM	Multi-Carrier Modulation.
MIMO	Multiple Input Multiple Output.
MISO	Multiple Input Single Output.
MMSE	Minimum Mean Square Error.
MSE	Minimum Square Error.
MMPP	Multidimensional Multicomponent Polynomial Phase.
MP	Matching Pursuit.
MPLS	Modified Predictive Least Square.
NMSE	Normalized Minimum Square Error.
NP	Non-deterministic Polynomial.

NTM	Non-deterministic Turing Machine.
NPH	NP-Hard.
OFDM	Orthogonal Frequency Division Multiplexing.
OMP	Orthogonal Matching Pursuit.
ORLS	Order Recursive Least Square.
PARAFAC	Parallel Factor.
PSO	Particle Swarm Optimization.
QAM	Quadrature Amplitude Modulation.
QPSK	Quadrature Phase-Shift Keying.
SAGE	Space Alternating Generalized Expectation.
SDET	Search, Detection, Estimation, and Tracking
ALS	Acoustic Linear Stochastic.
SFT	Short Fourier Transform.
SFTCE	Serial Frequency-Time Channel Estimator.
ISS	Imaging Sonar and Scattering.
SIMO	Single Input Multiple Output.
SISO	Single Input Single Output.
SNR	Signal to Noise Ratio.
SSIC	Soft Successive Interference Cancellation.
TFR	Time-Frequency Representations.
TM	Turing Machine.
TST	Tensor Space Time.
ToA	Time of Arrival.
URM	Unlimited Register Machine.
UTM	Universal Turing Machine.
UWA	Underwater Acoustic.
WSSUS	Wide-Sense Stationary Uncorrelated Scattering.
ZP	Zero Padding.

Notations

ϕ	Empty set
\mathbb{O}	Set with the null element $\{0\}$
\mathbb{Z}	Integer numbers set
\mathbb{Z}_N	Standard indexing set
\mathbb{K}	Scalar field
\mathbb{J}	Irrational numbers set
\mathbb{F}	Finite scalar field
\mathbb{P}	Prime numbers set
\mathbb{Q}	Rational numbers set
\mathbb{N}	Natural numbers set, where $\mathbb{N} = \{0, 1, 2, 3, \dots\}$
$\mathbb{N}_{>0}$	Non-negative integer numbers set, where $\mathbb{N}_{>0} = \{1, 2, 3, 4, \dots\}$
\mathbb{R}	Real numbers set
$\mathbb{R}_{>0}$	Positive real numbers set
$\mathbb{R}_{\geq 0}$	Non-negative real numbers set
\mathbb{C}	Complex numbers set
\mathbb{C}^N	Complex N -Dimensional vectors set
$\mathbb{C}^{M \times N}$	Complex $M \times N$ -Dimensional matrices set
x^*	Complex conjugated of signal $x \in l^2(\mathbb{Z})$
$ x $	Modulus of $x \in \mathbb{C}$
$\ x\ _2$	l_2 norm of $x \in l^2(\mathbb{Z})$
$\ y\ _2$	L_2 norm of $y \in L^2(\mathbb{R})$
$\ x\ _1$	l_1 norm of $x \in l^2(\mathbb{Z})$
$\ y\ _1$	L_1 norm of $y \in L^2(\mathbb{R})$
$\ x\ _\infty$	l_∞ norm of x
$\ y\ _\infty$	L_∞ norm of y
A^T	Transpose of matrix A
A^H	Hermitian of matrix A
A^{-1}	Inverse of matrix A
A^\dagger	Pseudo-inverse of matrix A $A^\dagger = (A^T A)^{-1} A^T$
$\mathbf{1}_N$	All-ones vector of dimension N
\mathbf{I}_N	Identity matrix of dimension $N \times N$

$\mathbf{A} \otimes \mathbf{B}$ Kronecker product of matrices $\mathbf{A} \in l^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and $\mathbf{B} \in l^2(\mathbb{Z}_K \times \mathbb{Z}_L)$

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{0,0}\mathbf{B} & a_{0,1}\mathbf{B} & \cdots & a_{0,N-1}\mathbf{B} \\ a_{1,0}\mathbf{B} & a_{1,1}\mathbf{B} & \cdots & a_{1,N-1}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M-1,0}\mathbf{B} & a_{M-1,1}\mathbf{B} & \cdots & a_{M-1,N-1}\mathbf{B} \end{bmatrix},$$

$$\mathbf{A} \otimes \mathbf{B} \in l^2(\mathbb{Z}_{MK} \times \mathbb{Z}_{NL})$$

$\mathbf{A} \overset{KR}{\otimes} \mathbf{B}$ Khatri-Rao Product

$$\mathbf{A} = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \\ A_{2,0} & A_{2,1} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} B_{0,0} & B_{0,1} \\ B_{1,0} & B_{1,1} \\ B_{2,0} & B_{2,1} \end{bmatrix},$$

$$\mathbf{A} \overset{KR}{\otimes} \mathbf{B} = \left[\begin{array}{c|c} A_{0,0} \otimes B_{0,0} & A_{0,1} \otimes B_{0,1} \\ \hline A_{1,0} \otimes B_{1,0} & A_{1,1} \otimes B_{1,1} \\ \hline A_{2,0} \otimes B_{2,0} & A_{2,1} \otimes B_{2,1} \end{array} \right],$$

$\mathbf{A} \oplus \mathbf{B}$ Direct sum of matrices $\mathbf{A} \in l^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and $\mathbf{B} \in l^2(\mathbb{Z}_K \times \mathbb{Z}_L)$

$$\mathbf{A} \oplus \mathbf{B} = \begin{bmatrix} \mathbf{A} & \mathbf{0}_{M \times L} \\ \mathbf{0}_{K \times N} & \mathbf{B} \end{bmatrix}, \mathbf{A} \oplus \mathbf{B} \in l^2(\mathbb{Z}_{M+K} \times \mathbb{Z}_{N+L})$$

$\mathbf{A} \sqcup \mathbf{B}$ Horizontal concatenation of matrices $\mathbf{A} \in l^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and $\mathbf{B} \in l^2(\mathbb{Z}_M \times \mathbb{Z}_L)$

$$\mathbf{A} \sqcup \mathbf{B} = [\mathbf{A} \mathbf{B}], \mathbf{A} \sqcup \mathbf{B} \in l^2(\mathbb{Z}_M \times \mathbb{Z}_{N+L})$$

$\mathbf{A} \vee \mathbf{B}$ Vertical concatenation of matrices $\mathbf{A} \in l^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and $\mathbf{B} \in l^2(\mathbb{Z}_K \times \mathbb{Z}_N)$

$$\mathbf{A} \vee \mathbf{B} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}, \mathbf{A} \vee \mathbf{B} \in l^2(\mathbb{Z}_{M+K} \times \mathbb{Z}_N)$$

$vec\{\mathbf{A}\}$ Vectorization operator applied to matrix $\mathbf{A} \in l^2(\mathbb{Z}_M \times \mathbb{Z}_N)$

$$vec\{\mathbf{A}\} = \begin{bmatrix} a_{0,0} \\ a_{1,0} \\ \vdots \\ a_{M-1,0} \\ \vdots \\ a_{0,N-1} \\ a_{1,N-1} \\ \vdots \\ a_{M-1,N-1} \end{bmatrix}, vec\{\mathbf{A}\} \in l^2(\mathbb{Z}_{MN})$$

\mathcal{B} Spline Interpolation Operator (IBC Context)

$\mathcal{C}\{v\}$ Circulant operator applied on vector $v \in l^2(\mathbb{Z}_M)$

$$\mathcal{C}\{v\} = \begin{bmatrix} v_0 & v_{M-1} & \dots & v_1 \\ v_1 & v_0 & \dots & v_2 \\ \vdots & \vdots & \ddots & \vdots \\ v_{M-1} & v_{M-2} & \dots & v_0 \end{bmatrix}, \mathcal{C}\{v\} \in l^2(\mathbb{Z}_M \times \mathbb{Z}_M)$$

$\mathcal{C}_h\{x\}$ Convolution operator applied on signal $x \in l^2(\mathbb{Z}_N)$

$$y = x \otimes h = \mathcal{C}_h\{x\}$$

$\mathbf{O}(f(n))$ Big O notation of the function $f(n)$

\mathbf{U} Delay-Doppler spread function

g Input-delay spread function

h Impulse response function

$\mathcal{Y}_{t_d, W}$ Windows-Delay-Reverse operator

$$\mathcal{Y}_{t_d, W} : l^2(\mathbb{Z}_M) \rightarrow l^2(\mathbb{Z}_W), t_d \in \mathbb{Z}_M, \text{ and } W \leq M$$

$$\mathcal{Y}_{0, W}\{x\} = [x_0, 0, 0, \dots, 0]$$

$$\mathcal{Y}_{1, W}\{x\} = [x_1, x_0, 0, \dots, 0]$$

$$\mathcal{Y}_{2, W}\{x\} = [x_2, x_1, x_0, \dots, 0]$$

$$\mathcal{Y}_{M-1, W}\{x\} = [x_{M-1}, x_{M-2}, x_{M-3}, \dots, x_{M-W}]$$

\mathcal{S}	Solution operator (IBC Context)
\mathcal{N}	Information operator (IBC Context)
\circ	Composition operator
\mathcal{D}	Time-Delay operator
\mathcal{M}	Multiplication operator
\mathcal{V}	Evaluation operator (IBC Context)
\mathcal{W}	Windowing operator
\mathcal{E}	Matrix reshape operator
\mathcal{K}	Sampling operator
\mathcal{X}	Sampling operator
\mathcal{O}	Any operator
\mathcal{R}	Reconstruction operator
Γ	Algorithm operator (IBC Context))
F	Problem space (IBC Context)
X	Information space (IBC Context)
G	Solution space (IBC Context)
Y	Evaluation space (IBC Context)

1. Introduction

The research work presented in this thesis is about new contributions to the theory of Computational Signal Processing (CSP), a unified branch of *signals theory* and *systems theory* which studies the computational treatment of signals, in order to extract relevant information important to an information user. A *signal* is defined in this work as any entity which is able of carrying information from one domain to another in *space-time*. We do not distinguish in this research work between a physical signal and its mathematical or model representation. The computational treatment of signals is carried out in this work by using computational methods. A computational method is defined as a non-empty structured set of computational tools utilized to solve mathematically posed problems. In this context, computational signal processing is a branch of computational complexity. Computational complexity is defined by J. F. Traub and A. Woźniakowski as the field which “*studies the intrinsic difficulty of mathematically-posed problems and seeks optimal means for their solution*” [1].

This thesis document describes new research work contributions in the area of *mathematical modeling* and presents research work results dealing with a novel formulation of a computational signal processing modeling framework designed and developed for the study of a generalized integrated Multiple-Input Multiple-Output (MIMO) Stochastic-Acoustic-Linear (ALS) communications and *imaging-search-sonar* (ISS) operations systems, for underwater applications, which we have named ALSISS, and its associated computational complexity.

My research work concentrated on the study of intrinsic difficulties encountered when trying to obtain approximated solutions to optimal computational methods associated with the proposed modeling framework which was mathematically posed

in order to be treated with targeted analytics tools from the field of computational complexity. My research work dealt, in particular, with a branch of computational complexity known as information-based complexity (IBC). As defined by Wershulz, A.G., IBC is a discipline which studies *“the complexity of problems for which the available information is partial, contaminated by error [or interference], and priced”* [2] [3].

The major contribution of my work is the development of this computational modeling framework which can be utilized to study mathematically posed problems which deal with infinite-dimensional continuous signal spaces contaminated with interference or noise signals. These signal spaces are acted upon by [complete] linear operators representing computational systems which are used to estimate relevant signals and parameters important to an information user.

The computational methods utilized to address the formulated estimation problems tend to be NP-complete. For this reason in this work we studied error approximation computational methods. In this context, a computational method may be defined as a structured set of computational tools.

1.1 Justification & Problem Formulation

The main aspects of the research work addressed in this thesis have their origins in an early concern of mine when I began to conduct research in acoustic signal processing. I was concerned about how to seek solutions to problems pertaining to underwater environmental surveillance and monitoring activities in shallow lakes and marine ecosystems (0 to 100 meters in depth) using acoustic signals. In particular, I was concerned about environmental surveillance and monitoring activities in Lake Maracaibo.



Figure 1–1: Satellite view of Lake Maracaibo. Source: ICLAM

Lake of Maracaibo is a somewhat large and slightly salty lake situated in the northwestern part of Venezuela, my native country, in South America. Lake Maracaibo is considered to be the principal component of the known Maracaibo Basin, an extremely rich hydrocarbon producing region of the Earth, with an estimated $1.5 \times 10^9 m^3$ of oil reserves. The lake is about 122 kilometers long and 110 kilometers wide, with an area of about 13,420 square kilometers. By comparison, the Commonwealth of Puerto Rico has an estimated area of about 13,790 square kilometers, for the whole archipelago. The main island of Puerto Rico, considered the smallest and most eastern of the Greater Antilles, has an area of 9,100 square kilometers. Figure 1–1 shows the bathymetric information about the Lake Maracaibo system, and Figure 1–2 shows a satellite view of Lake Maracaibo. These images were offered by the Information Systems Department of the *Conservation Lake Maracaibo Institute* (ICLAM for its acronym in Spanish).

Lake Maracaibo is connected on the north to the Gulf of Venezuela and the Caribbean Sea through a narrow water channel, that is about 14 meters in depth

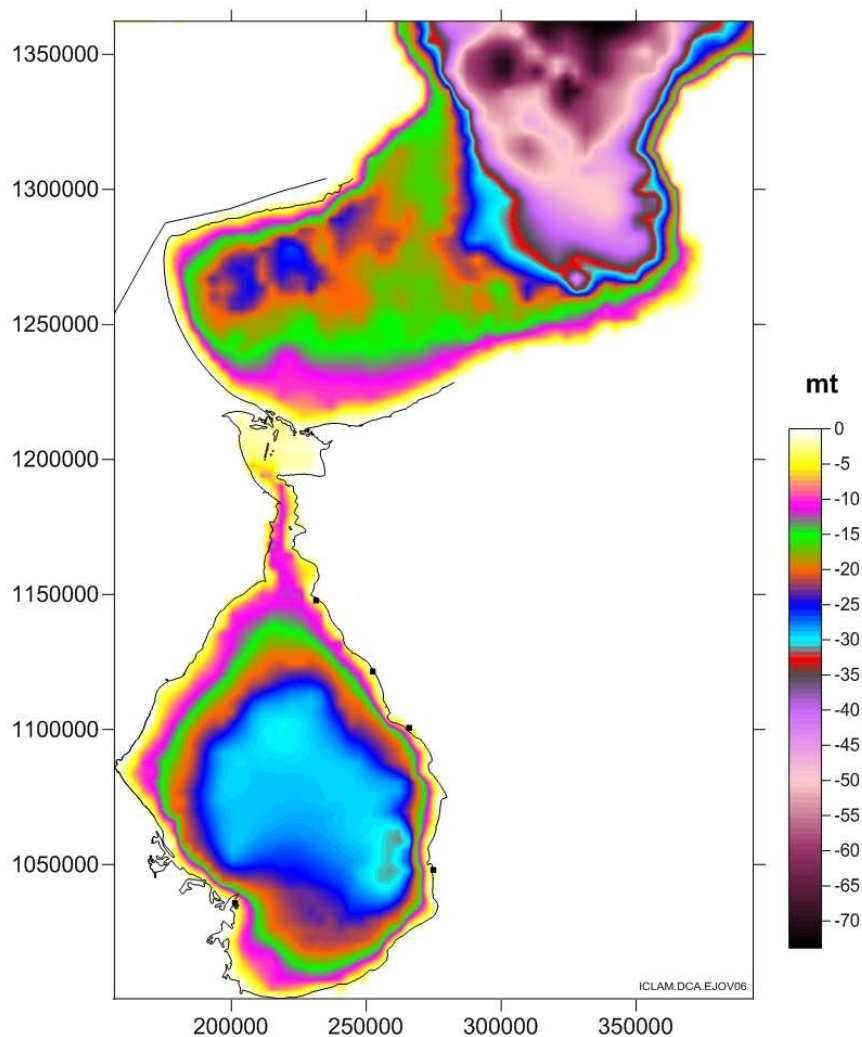


Figure 1–2: Graphical Representation of the Bathymetry of Lake Maracaibo system.
Source: ICLAM

and about 200-300 meters wide. This narrow channel is called Maracaibo Strait and runs through Tablazo Bay. The lake is a basin like lake, with its deepest part estimated at about 60 meters. It is considered one of the oldest lakes on Earth. Lighter fresh water comes from many river tributaries, its main tributary being the Catatumbo River, and floats on top of heavier salt water which comes from the Caribbean Sea. Over a narrow region of the Tablazo Bay, across the Maracaibo Strait, spans the great bridge named *General Rafael Urdaneta*. The bridge is about 8.68 kilometers long.



Figure 1-3: Oil Rigs in Lake Maracaibo



Figure 1-4: Hundreds of derricks pump oil from Lake Maracaibo.

Grave environmental situations and concerns about the current condition of the lake are demanding new approaches at addressing these concerns. One of these new approaches deals with acoustic environmental monitoring.

We are interested in searching and gathering new information, knowledge, and understanding from data gathered in Lake Maracaibo through acoustic underwater *environmental surveillance and monitoring* (eSAM) operations. The undertaking of acoustic communications in any underwater environment becomes a very difficult

enterprise since the underwater propagation of an acoustic wavefront experiences undesirable changes due primarily to two significant reasons: time-frequency acoustic signal transformations and excessive signal multipath effects. An additional factor which contributes to the great difficulty encountered when trying to communicate through an underwater acoustic medium is the low speed of sound propagation in water. Even though, at a rate of about 1,560 meters per second, the speed of the sound in the water is much faster than acoustical energy moving through the air, at a rate of about 340 meters per second, such speed in water is extremely low when compared with electromagnetic energy propagation at rate of 300,000,000 meters per second.

In this work, I concentrate on the study of the problem of the time-frequency transformations experienced by an acoustic waveform underwater signal and how to develop computational signal processing methods to minimize these adverse transformations in processes pertaining to the search, detection, estimation, and tracking (termed SDET processes) of underwater acoustic signals. A whole chapter of this thesis is dedicated to the theory of time-frequency signal analysis, where we present some original contributions. Another whole chapter is also dedicated to the formulation of a Computational Signal Processing (CSP) modeling framework, where systems are formulated and integrated in an unified manner to model certain computational aspects of these SDET processes.

1.2 Problem Space

In this thesis work *problem space* denotes the signal space where all signals that are object of consideration inside the problem are encountered. This thesis work addresses the fundamental question of how to characterize randomly time-variant channels when subjected to a special class of analytic signals as inputs; in particular, the class of Multidimensional, Multicomponent, Polynomial Phase (MMPP) signals.

In this work, an analytic signal, implies the analytic representation of a real-valued function, assuming *hermitian symmetry*. In general, an analytic signal or function is any function which can always be locally described by a convergent power series. Particular emphasis is given in this work to the characterization of Acoustic Linear Stochastic (ALS) channels for broadband Multiple-Input Multiple-Output (MIMO) multi-carrier communication.

Let $\mathbf{x}(t)$ be an MMPP analytic input signal to an ALS channel. Let $\mathbf{y}(t) = \mathbf{x}(t) + \mathbf{n}(t)$ be the output signal of the channel, where $\mathbf{n}(t)$ is a stochastic signal representing the additive white Gaussian noise exhibited in the channel. A typical example of this type of signals are the *chirp signals*, whose corresponding time-domain function, for a sinusoidal linear case, is the sine of the phase in radians:

$$x(t) = \sin \left[\phi_0 + 2\pi \left(f_0 t + \frac{k}{2} t^2 \right) \right], \quad (1.1)$$

where the instantaneous frequency is described by the equation

$$f(t) = f_0 + kt, \quad (1.2)$$

and it is accompanied by the harmonics that appear due to frequency modulation.

Problem Statement: How to formulate, in an integrated manner, the complexity associated with the characterization of the impulse response function of the channel, denoted by $\mathbf{h}(t, \tau)$, in order to address applications in three specific areas:

- i. **Performance evaluation of digital underwater acoustic communications (a *detection & estimation* problem), and**
- ii. **Description of underwater moving objects (a *SONAR* problem).**

We start by modeling the signal-channel interaction in the following manner:

$$\mathbf{y}(t) = \int \mathbf{h}(t, \tau) \mathbf{x}(t - \tau) d\tau + \mathbf{n}(t). \quad (1.3)$$

Using $\mathbf{U}(\nu, \tau)$ as the Fourier transform of the impulse response function or object-domain point spread function $h(t, \tau)$, we may describe the above channel input/output model as follows:

$$\iint \mathbf{U}(\nu, \tau) \mathbf{x}(t - \tau) e^{j2\pi\nu t} d\tau d\nu + \mathbf{n}(t). \quad (1.4)$$

The function $\mathbf{U}(\nu, \tau)$ is usually termed the *Delay-Doppler Spread Function* of the ALS channel. This function is of fundamental importance in our research work since it allowed to formulate our solution under a time-frequency signal representation computational framework. Time-frequency signal representation theory is a well established discipline. However, most important and fundamental results are provided in analytic form, with very few results presented under a *computational setting*. By a computational setting we imply any formulation or set of formulations which may readily admit an algorithmic implementation in a digital computer. Our solution approach provided an original contribution by formulating desired results in a time-frequency computational modeling framework, using the discrete Cohen class of time-frequency distributions as our main computational tools which will be defined in the next section. In particular, we concentrated on the discrete Ambiguity function as the basis for an unique discrete formulation of the Cohen class of time-frequency distributions, following the work of J.P. Soto and D. Rodriguez [4].

In 1953, P.M. Woodward developed the concept of Ambiguity Function. The Ambiguity Function (AF) plays an important role in many applications dealing with the analysis of non-stationary signals [5, 6]. It is finding new roles in applications such as the joint time-frequency analysis of Multiple-Input Multiple-Output (MIMO) doubly dispersive channels and phase-coded waveform design for orthogonal frequency division multiplexing (OFDM) radar sensing and communications. L. Cohen developed the general representation of a signal in continuous time and frequency (*two dimensional distribution*) calling it “*general class*” (GC).

1.2.1 Continuous Ambiguity Functions and the General Class of Cohen Distributions

The *Continuous Ambiguity Function* of $x, y \in L^2(\mathbb{R})$ is defined as a map

$$A_{x,y} : \mathbb{R} \times \mathbb{R} \rightarrow L^2(\mathbb{R} \times \mathbb{R}), \quad (1.5)$$

is given by

$$A_{x,y}(\tau, \nu) = e^{i\pi\nu\tau} \int x(t + \tau) y^*(t) e^{j2\pi\nu t} dt. \quad (1.6)$$

In the case when $x = y$, $A_{x,y}$ becomes A_x . L. Cohen [7, 8] developed the concept of a *general class* for two dimensional distributions. As described by L. Cohen, all time-frequency representations, for $x \in L^2(\mathbb{R})$, can be obtained from the canonical map

$$C_x : \mathbb{R} \times \mathbb{R} \rightarrow L^2(\mathbb{R} \times \mathbb{R}), \quad (1.7)$$

expressed as

$$C_x(t, \omega) = \int \int A_x(\tau, \eta) \phi(\tau, \eta) e^{-j2\pi(t\eta + \omega\tau)} d\eta d\tau, \quad (1.8)$$

where the function $\phi \in L^2(\mathbb{R} \times \mathbb{R})$ is termed the *kernel of the representation*. Previous studies have been conducted for the general class of two dimensional distributions using specific kernel functions. Some known kernel functions are presented in Table 1, below.

Distribution	$\phi(\tau, \theta)$	$\phi(\tau, \eta)$
Wigner	1	1
Margenau - Hill	$\cos(\tau\theta/2)$	$\cos(\pi\tau\eta)$
Kirwood-Rihanzek	$e^{-\tau\theta/2}$	$e^{-\tau\pi\eta}$
Born - Jordan	$\frac{\sin(\tau\theta/2)}{\tau\theta/2}$	$\frac{\sin(\pi\tau\eta)}{\pi\tau\eta}$
Choi - Williams	$e^{-\alpha(\tau\theta)^2}$	$e^{-4\alpha(\pi\tau\eta)^2}$
Zhao-Atlas-Marks	$e^{-\alpha\tau^2} \tau \frac{\sin(\alpha\theta\tau)}{\alpha\theta\tau}$	$e^{-\alpha\tau^2} \tau \frac{\sin(2\pi\alpha\eta\tau)}{2\pi\alpha\eta\tau}$

Table 1: Commonly Known Kernel Functions

1.2.2 Discrete Ambiguity Functions (DAF) and Discrete Cohen Distributions (DCD)

In [9], M. Richman, et al., presented a discrete formulation for the expression given in (1.6). This new expression is called the *Discrete Ambiguity Function* (DAF). The DAF, for $x, y \in l^2(\mathbb{Z}_N)$, defined by the map

$$A_{x,y} : \mathbb{Z}_N \times \mathbb{Z}_N \rightarrow l^2(\mathbb{Z}_N \times \mathbb{Z}_N), \quad (1.9)$$

is given by

$$A_{x,y}[\tau, \nu] = \rho_{\tau,\nu} \sum_{l=0}^{N-1} x[\langle l + \tau \rangle_N] y^*[l] e^{j\frac{2\pi}{N}\nu l}, \quad (1.10)$$

where we call the expression $\rho_{\tau,\nu}$ *the ambiguity function's phase factor*, and it is defined in [9]. M. Richman, et al., proceeded to use finite dimensional linear operators to arrive at an operator formulation of the DAF. The procedure is as follows:

For $x \in l^2(\mathbb{Z}_N)$, $\gamma, \mu \in \mathbb{Z}$ and $n \in \mathbb{Z}_N$, we define the following linear operators:

- *Translation*: $S_\gamma\{x\}[n] = x[\langle n + \gamma \rangle_N]$
- *Modulation*: $M_\mu\{x\}[n] = e^{j\frac{2\pi}{N}\mu n} x[n]$

For $x, y \in l^2(\mathbb{Z}_N)$, $\langle x, y \rangle$ denotes their inner product and is given by $\langle x, y \rangle = \sum_{n \in \mathbb{Z}_N} x[n]y^*[n]$. Using these definitions, the operator formulation for the DAF follows from (1.10):

$$A_{x,y}[\tau, \nu] = \rho_{\tau,\nu} \langle M_\nu \{S_\tau \{x\}\}, y \rangle. \quad (1.11)$$

This operator formulation of DAF has served as a point of inspiration to develop a general class of Discrete Cohen Distributions (DCD), expressed in canonical form in the following manner: for $x \in l^2(\mathbb{Z}_N)$, define the map

$$C_x : \mathbb{Z}_N \times \mathbb{Z}_N \rightarrow l^2(\mathbb{Z}_N \times \mathbb{Z}_N), \quad (1.12)$$

as the following expression,

$$C_x[n, k] = \frac{1}{N} \sum_{\tau=0}^{N-1} \sum_{\nu=0}^{N-1} \rho_{\tau,\nu} \langle M_\nu \{S_\tau \{x\}\}, x \rangle \cdot \phi[\tau, \nu] e^{-j \frac{2\pi}{N} (n\nu + k\tau)}, \quad (1.13)$$

where ϕ is the kernel in $l^2(\mathbb{Z}_N)$. When $\phi[\tau, \nu] = 1$, for $\tau, \nu \in \mathbb{Z}_N$, the DCD reduces to the discrete Wigner distribution (DWD) as defined in [10].

Let $\mathcal{F} : \mathbb{Z}_N \times \mathbb{Z}_N \rightarrow l^2(\mathbb{Z}_N \times \mathbb{Z}_N)$ be the symmetric discrete Fourier transform in two dimensions (2D); then, (1.13) defines the symmetric discrete Fourier transform, in 2D, of the product of the DAF and the kernel [9]; i.e.,

$$C_x[n, k] = \mathcal{F}\{A_x \cdot \phi\}[k, n]. \quad (1.14)$$

1.2.3 Relating DAF and DCD Operations

The concept of the Discrete Cohen Distributions of a signal was introduced in the previous section through equation (1.13). In this section we re-formulate properties of the DCD based on J. J. Benedetto and J. J. Donatelli [11], who presented the following theorem for the DAF of a signal after being effected by translation, modulations, and rotations operators. The original formulations of these properties appear in [10]. We first introduce the definition of the rotation operator: For

$x \in l^2(\mathbb{Z}_N)$, $\lambda \in \mathbb{C}$, with $|\lambda| = 1$, and $n \in \mathbb{Z}_N$, the rotation operator $R_\lambda\{x\}$ is defined as

$$R_\lambda\{x\}[n] = \lambda x[n]$$

The translation $S_{-\gamma}\{x\}$ and modulation $M_\mu\{x\}$ operators were defined in the previous section.

Theorem 1 *Let $x \in l^2(\mathbb{Z}_N)$, $\gamma, \mu, \lambda \in \mathbb{Z}$, with $|\lambda| = 1$, and $\tau, \nu \in \mathbb{Z}_N$:*

1. $A_{S_{-\gamma}\{x\}}[\tau, \nu] = e^{-j\frac{2\pi}{N}\gamma\nu} A_x[\tau, \nu]$
2. $A_{M_\mu\{x\}}[\tau, \nu] = e^{j\frac{2\pi}{N}\mu\tau} A_x[\tau, \nu]$
3. $A_{R_\lambda\{x\}}[\tau, \nu] = A_x[\tau, \nu]$

From **Theorem 1**, and equation (1.14), we obtain the following proposition:

Proposition 1 *Let $x \in l^2(\mathbb{Z}_N)$ and $\gamma, \mu, \lambda, n, k \in \mathbb{Z}_N$, with $|\lambda| = 1$:*

1. $C_{S_{-\gamma}\{x\}}[n, k] = C_x[\langle n + \gamma \rangle_N, k]$
2. $C_{M_\mu\{x\}}[n, k] = C_x[n, \langle k - \mu \rangle_N]$
3. $C_{R_\lambda\{x\}}[n, k] = C_x[n, k]$

1.3 Proposed Solutions

The solutions proposed in this thesis work are based on the integration of two computational signal processing frameworks (ALS & ISS) as a composed framework (ALS-ISS) enabled to model, with significant level of accuracy, the behavior of an underwater acoustic linear stochastic channel, in the context of the communication and sonar problems. For this purpose the author used time-frequency analysis tools as solid mathematical foundations that allowed the construction of two computational frameworks. A valued perspective was added to the work performing an *information-based complexity* analysis of the computational frameworks.

1.4 Thesis Organization

This thesis document was organized as follow: this chapter addresses the introductory concepts and the justification for this research. Chapter 2 elaborates a compact theoretical foundation survey, about the elementary concepts and definitions treated in this thesis work. Chapter 3 presents a background review of 3 journal papers related to main topics in this research. Chapter 4 introduces the Information-Based Complexity (IBC) concepts and relates these concepts with the Computational Signal Processing Theoretical Framework (CSP) concepts presented in this thesis. Important contributions are presented in Chapter 4. Chapter 5 addresses the Acoustic Linear Stochastic (ALS) systems theory. Chapter 6 develops the parameter estimation matching pursuit algorithms. This last chapter is fundamental since it presents the Single Input Single Output (SISO), Single Input Multiple Output (SIMO), Multiple Input Single Output (MISO), and Multiple-Input Multiple-Output (MIMO) approaches related to the parameter estimation problem in the ALS channels context. Chapter 7 presents some ethical considerations related to the main research topic. Finally, Chapter 8 presents conclusions and shortly describes future works.

2. Elementary Concepts and Ideas

2.1 Introduction

In this section we describe important concepts related to this research. These concepts form the theoretical foundation to begin a systematic search of knowledge that would enable us to successfully reach conclusions solidly grounded.

2.2 Kronecker Products

Let $A \in l^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and $B \in l^2(\mathbb{Z}_Q \times \mathbb{Z}_R)$ be two matrices. The Kronecker product of A and B is a binary operation which results in a new matrix $C \in l^2(\mathbb{Z}_{MQ} \times \mathbb{Z}_{NR})$, denoted by $C = A \otimes B$, given by

$$C = A \otimes B = \begin{bmatrix} a_{0,0}B & a_{0,1}B & \dots & a_{0,N-1}B \\ a_{1,0}B & a_{1,1}B & \dots & a_{1,N-1}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{M-1,0}B & a_{M-1,1}B & \dots & a_{M-1,N-1}B \end{bmatrix}. \quad (2.1)$$

If the matrix A is equal to the identity matrix I of order M , then

$$C = I_M \otimes B = \underbrace{\begin{bmatrix} B & \mathbf{0}_{Q \times R} & \mathbf{0}_{Q \times R} \\ \mathbf{0}_{Q \times R} & B & \dots \mathbf{0}_{Q \times R} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{Q \times R} & \mathbf{0}_{Q \times R} & \dots & B \end{bmatrix}}_M. \quad (2.2)$$

The equation (2.2) can be expressed by a *direct sum* of matrices as follow

$$C = \bigoplus_{i \in \mathbb{Z}_M} (B) = \begin{bmatrix} B & \mathbf{0}_{Q \times R} & \mathbf{0}_{Q \times R} & \\ \mathbf{0}_{Q \times R} & B & \dots \mathbf{0}_{Q \times R} & \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{Q \times R} & \mathbf{0}_{Q \times R} & \dots & B \end{bmatrix}. \quad (2.3)$$

The Kronecker product is not a commutative operator. If,

$$A \otimes B = \begin{bmatrix} a_{0,0}B & \dots & a_{0,N-1}B \\ \vdots & \ddots & \vdots \\ a_{M-1,0}B & \dots & a_{M-1,N-1}B \end{bmatrix}, \quad (2.4)$$

and,

$$B \otimes A = \begin{bmatrix} b_{0,0}A & \dots & b_{0,R-1}A \\ \vdots & \ddots & \vdots \\ b_{Q-1,0}A & \dots & b_{Q-1,R-1}A \end{bmatrix}, \quad (2.5)$$

then,

$$A \otimes B \neq B \otimes A. \quad (2.6)$$

2.2.1 Kronecker Products and Parallel Formulations

Any expression of the form $I_M \otimes X_N$ can be seen as a parallel operation since the nonzero elements, the matrices X_N , appear along the diagonal. The matrix $I_M \otimes X_N$ is a sparse matrix and its implementation favors a parallel architecture. This can be demonstrated with a little example. Take $M = 3$ and $N = 2$. If we

compute $y = (I_M \otimes X_N)h$, this matrix-vector multiplication operation becomes

$$\begin{bmatrix} y_0 \\ y_1 \\ \hline y_2 \\ y_3 \\ \hline y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} x_{0,0} & x_{0,1} & 0 & 0 & 0 & 0 \\ x_{1,0} & x_{1,1} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & x_{0,0} & x_{0,1} & 0 & 0 \\ 0 & 0 & x_{1,0} & x_{1,1} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & x_{0,0} & x_{0,1} \\ 0 & 0 & 0 & 0 & x_{1,0} & x_{1,1} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \hline h_2 \\ h_3 \\ \hline h_4 \\ h_5 \end{bmatrix} = \begin{bmatrix} x_{0,0}h_0 + x_{0,1}h_1 \\ x_{1,0}h_0 + x_{1,1}h_1 \\ \hline x_{0,0}h_2 + x_{0,1}h_3 \\ x_{1,0}h_2 + x_{1,1}h_3 \\ \hline x_{0,0}h_4 + x_{0,1}h_5 \\ x_{1,0}h_4 + x_{1,1}h_5 \end{bmatrix}. \quad (2.7)$$

If the column vector h is divided into 3 sections, being of length 2 each, the computation $y = (I_3 \otimes X_2)h$ could be performed by computing three simultaneous blocks of length 2. In the general form, operation $y = (I_M \otimes X_N)h$ can be performed by computing M simultaneous blocks of length N each one. If we have a computer architecture with M processors, then these operations could be performed concurrently.

2.2.2 Kronecker Products and Vector Formulations

The kronecker product $X_M \otimes I_N$ also has special properties. This expression favors an architecture with vector processing capabilities. The operation $y = (X_M \otimes I_N)h$ can be computed at a vector level instead of at a scalar level as can be demonstrated by the following example.

Let $M = 2$ and $N = 4$. Then, the operation $y = (X_2 \otimes I_4)h$ becomes

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \left(\begin{bmatrix} x_{0,0} & x_{0,1} \\ x_{1,0} & x_{1,1} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \end{bmatrix}. \quad (2.8)$$

After performing the Kronecker product we obtain

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} x_{0,0}I_4 & x_{0,1}I_4 \\ x_{1,0}I_4 & x_{1,1}I_4 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \end{bmatrix} = \begin{bmatrix} x_{0,0} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} + x_{0,1} \begin{bmatrix} h_4 \\ h_5 \\ h_6 \\ h_7 \end{bmatrix} \\ x_{1,0} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} + x_{1,1} \begin{bmatrix} h_4 \\ h_5 \\ h_6 \\ h_7 \end{bmatrix} \end{bmatrix}. \quad (2.9)$$

This matrix-vector multiplication operation can be seen as a submatrix-vector segment multiplication. In general, the operation $y = (X_M \otimes I_N)h$ can be computed into a machine with vector processing capabilities. The input vector could be divided into M segments of length N each one. The expressions $I_M \otimes X_N$ and $X_M \otimes I_N$ are used in the Kronecker product formulations of *FFT algorithms* and become instrumental in obtaining efficient *FFT implementations*.

2.3 Modeling and Simulation

We can begin defining what is a model. A model is a simplified representation of a system. This representation is focused in some particular point, in time and space, allowing to understand or predict the real system. Now, what is a simulation? A simulation is the manipulation of a model, changing their spatial and temporal parameters in order to assess its performance. In this way, we can appreciate the relationship between the variation of its parameters and to infer possible interactions between them. Modeling and Simulation is a discipline for developing a level of understanding of the interaction of the parts of a system, and of the system as a whole [12].

A system is an entity which shows its existence through the interaction of its parts. A model is a simplified representation of the actual system, developed to facilitate its understanding. A model abstracts the entity it represents, ignoring certain details that are not relevant to the perspective that you want to highlight. Since all models are simplifications of reality, there is always a trade-off as to what level of detail is included in the model. Bellinger referring about risk of models say: *“If too little detail is included in the model one runs the risk of missing relevant interactions and the resultant model does not promote understanding. If too much detail is included in the model the model may become overly complicated, making impossible its implementation”* [12].

A simulation, generally refers to a computerized version of the model which is run over time, to study the implications of the defined interactions. Simulations are generally iterative in their development. One develops a model, simulates it, learns from the simulation, revises the model, and continues the iterations until an adequate level of understanding is developed.

It has been said that “*modeling and simulation is a discipline and an art. From the interaction of the developer and the model emerges an understanding of what makes sense and what doesn't,*” [12].

A Computational Signal Processing (CSP) modeling framework is an aggregate of the following components:

- A set of input signals
- A set of output signals
- A set of linear operators
- A set of composition rules for the linear operators
- A set of action rules for the linear operators
- A user interface.

When we are dealing with finite dimensional linear operators, the CSP modeling framework may be implemented under a computational matrix algebra environment or a *signal algebra environment*, in general. We used the numeric computational and visualization package MATLAB to develop an instantiation of our proposed CSP modeling framework.

2.4 The Stochastic Processes

A continuous, t-dependent, stochastic process is a family of random variables $x = \{^t\xi; t \in \mathbb{R}\}$. We must recall that a random variable $^t\xi$ is a measurable function:

$$\begin{aligned} ^t\xi(\omega) : \Omega &\longrightarrow \mathbb{R} \\ \omega &\longmapsto ^t\xi(\omega) = {}^t x_0 = {}^t\xi_0, \end{aligned} \tag{2.10}$$

defined on a probability space $(\Omega, \mathbb{S}, \mathbf{P})$, where for each $\omega \in \Omega$, we have the function:

$$\begin{aligned} b_\omega : \mathbb{R} &\longrightarrow \mathbb{R} \\ t &\longmapsto b_\omega(t) = {}^t\xi(\omega), \end{aligned} \tag{2.11}$$

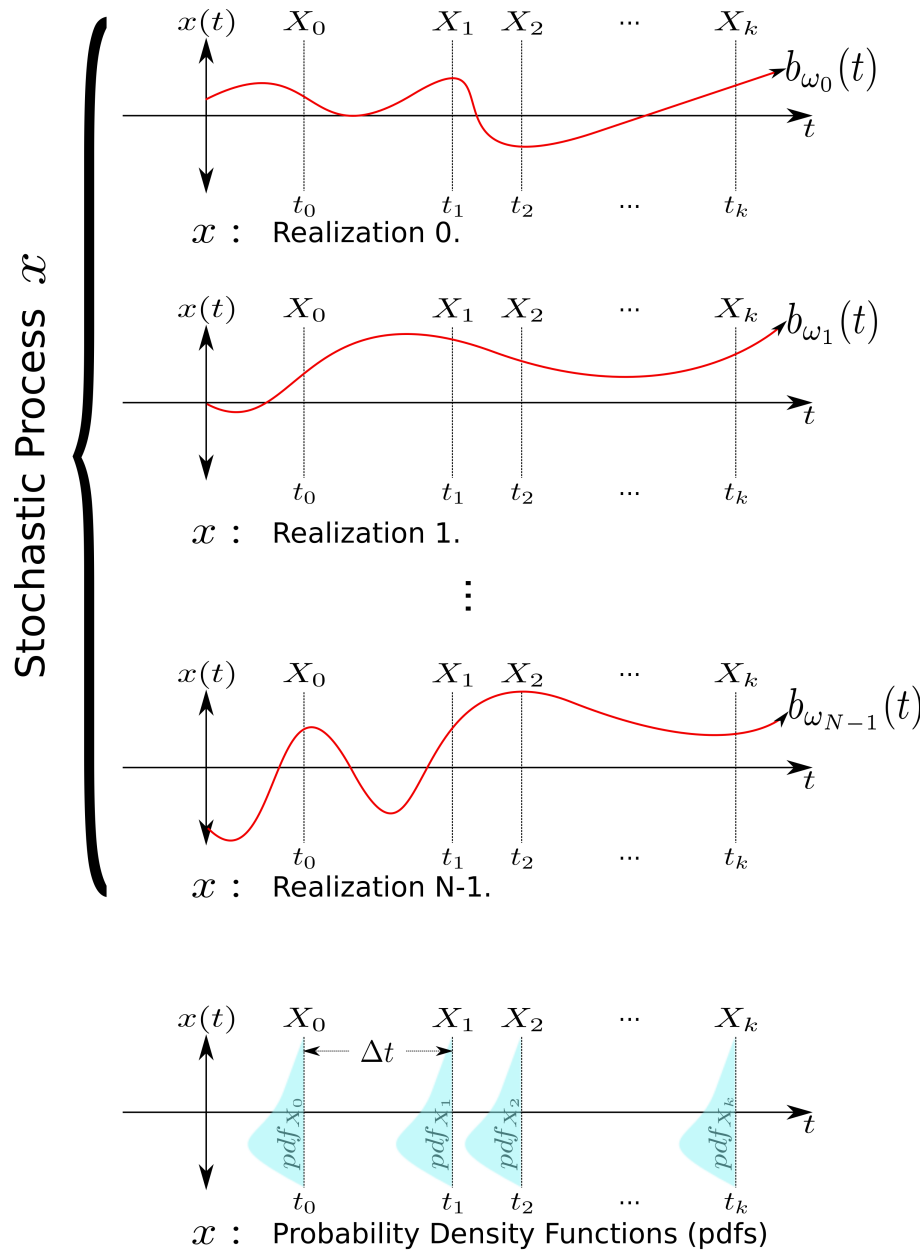
is called the *orbit* or *trajectory* of the process.

The random distribution of any signal (random vector) \mathbf{x} can be considered as a stochastic process with co-domain in the space of probability measures $\mathbb{X} \subseteq \mathbb{R}$. A meaningful problem consists in determining the random distribution of a signal $x(t)$ given the observation of its σ -algebra \mathbb{S}_t .

2.5 Random Fields

From a mathematical perspective, *random fields* constitute a branch that studies random (nondeterministic) functions. *Random fields* is considered part of *functional analysis* and deals with topics covered in *theory of functions*. The existence of a random component transforms functional analysis in a more wide discipline, more complex, and more challenging.

In this research work we are dealing with estimation of stochastic linear time-variant channels. So, *Random fields* theory will be used in order to deal with the stochastic estimation processes. Figure 2-2 shows all mathematic resources used in *random fields theory* [13].



Stochastic Process	Statistic Conditions
Ergodic	Statistics of x equal to Statistics of each Stochastic Variable X_k (Perfect Match Statistics)
Stationary	Statistics of each Stochastic Variable X_k equal to each (Time Invariant Statistics)
Wide Sense Stationary	Statistics of each Stochastic Variable X_k depends of Δt (Pseudo Time Invariant Statistics)

Figure 2-1: Stochastic Process x with its Stochastic Variables X_k .

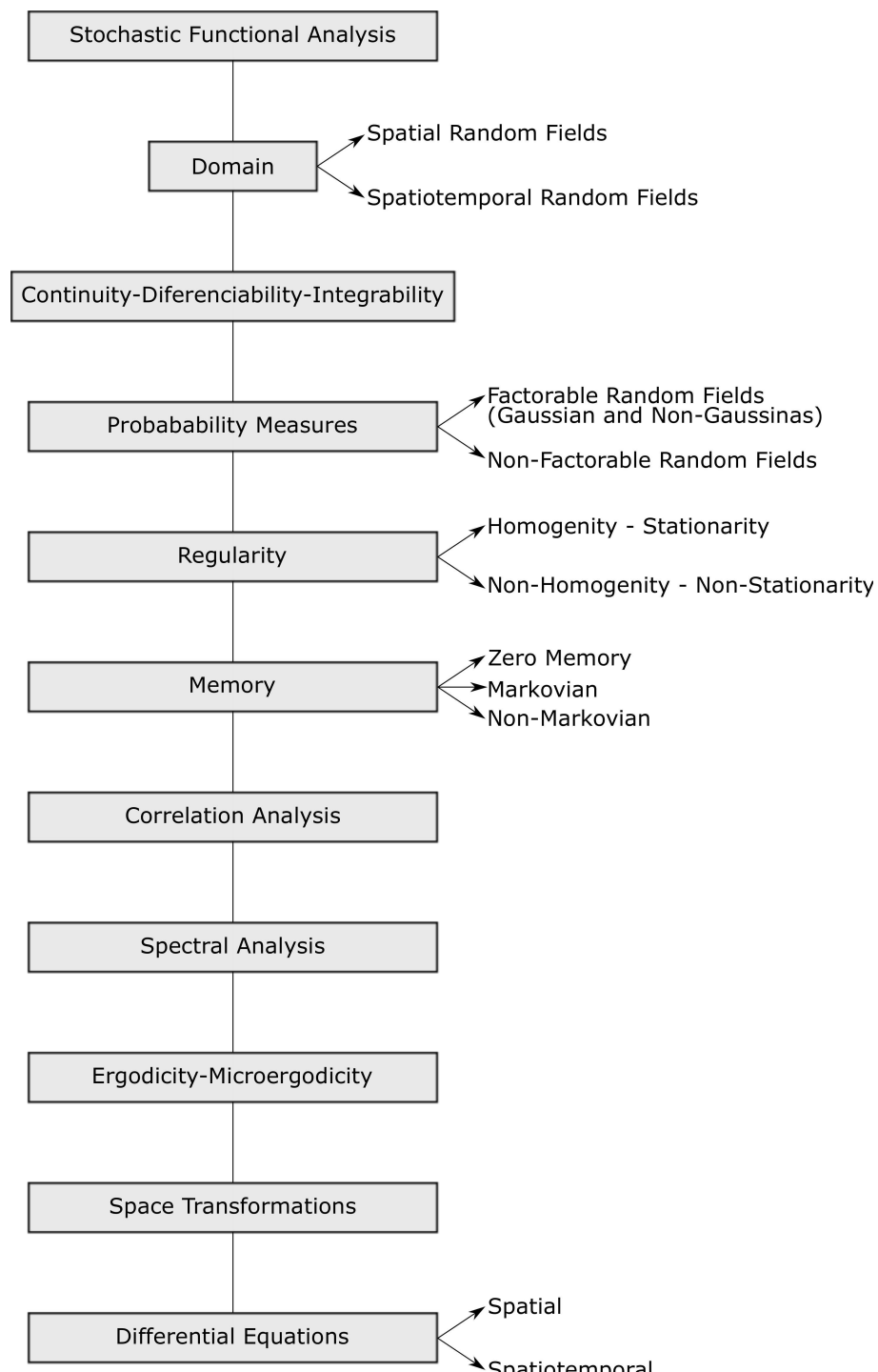


Figure 2–2: George Christakos showed in his work “*Random Fields Models In Earth Sciences*” which are the Mathematics Behind the Random Fields Theory

3. Background and Related Works

3.1 Literature Review

3.1.1 Introduction

In this section we describe important contributions which are related to the main themes presented in the nature of the proposed work. A total of nineteen (19) research contributions are discussed, ranging from the time-frequency analysis of doubly-dispersive ALS channels to information-base complexity associated with dynamical systems. Each article contribution is discussing and addressing four relevant issues: (1) problem formulation, (2) theoretical framework, (3) mathematical and computational resources, and (4) how this work is related to my thesis research.

3.1.2 Characterization of Randomly Time-Variant Linear Channels

The characterization of Linear Time-Invariant (LTI) systems is relatively easy [14]. But, most systems in the practical world are not LTI. The linear condition can be approximated; however, many systems with practical interest are usually modeled as linear time variant systems. So, characterization of random time-variant linear systems (like channels) is very important in scientific and engineering applications. Wide-Sense Stationary Uncorrelated Scattering (WSSUS) channels have gained much attention and they represent a very important type of practical systems; so, in this article they are studied deeply. The kernel definition is used for signal characterization under an integral operator context. Systems theory, stochastic methods, and filter theory are often used in this reviewed work. The channels that were considered in our research work are stochastic channels and their modeling must be supported by using randomly time-variant linear channel theory.

This paper represents a pioneering effort to analyze in a formal manner the behavior of the time variant communications systems [15]. The linear time-invariant systems theory was well known at that time; however, the characterization of time variant systems was poorly understood in 1963. That year, P. Bello presented this magisterial paper about characterization of randomly time-variant linear channels. This paper has been used as reference work in countless papers in the communication and systems branches. This paper was analyzed in detail to fully understand the basis of time-variant channel characterization. This paper presents the main system functions used as characterization tools for communication channels, establishing mathematical relations among these functions. Fourier theory is a fundamental mathematical analysis tool in this paper to establish this relations. Each function is analyzed in a dual manner, both, in the time domain and frequency domains.

This paper demonstrates that time-variant linear channels may be characterized in a dual symmetrical manner in the time and frequency domains. Most of

the system functions are discussed making use of circuit model interpretations or representations for the time-variant linear channels. The relationship between these system functions is demonstrated in a simple way with the aid of block diagrams involving time-frequency duality and Fourier transformations. Other important contribution of this paper is the establishing of relationships between the correlation functions of the various system functions for the general randomly time-variant linear channel. This paper addresses three classes of channels of theoretical as well as practical interest: the WSS channel, the US channel, and the WSSUS channel. The WSS and US channels are shown to be (time-frequency) duals.

Relation to my thesis work: My thesis work is centered on the study of MIMO ALS channels. This type of channel can be categorized as time-variant double dispersive channels; so, this paper is extremely important in establishing the foundations to develop a useful and accurate model. Most of the time-frequency theory discussed by the author in this paper is employed in my thesis work to create an analytic model for acoustic stochastic linear (ALS) time-variant systems. This analytic model is converted into a computational model using finite-dimensional linear operator theory and other mathematical and computational resources. In this perspective, this paper established the essential foundation of my research.

Figure 3–1 illustrates, in block diagram form, the transformation processes between different representations of the system functions for time-variant linear channels [15].

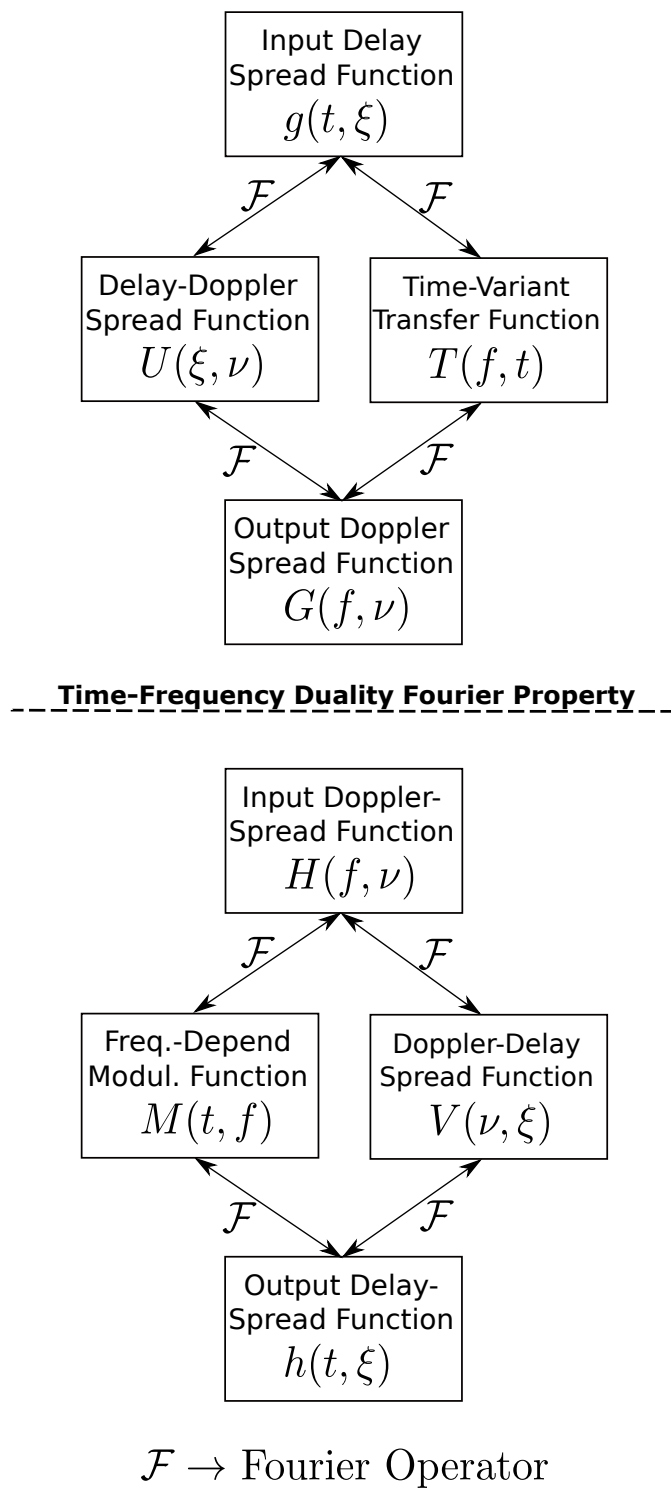


Figure 3-1: Relationship Between System Functions in Time-Variant Channels

3.1.3 Information-Based Complexity

This paper is about *Information-based Complexity* (IBC) and how IBC “seeks to develop general results about the intrinsic difficulty of solving problems where available information is partial or approximate and to apply these results to specific problems,” [16].

The field of complexity theory has improved many performance measurements on the Turing and non-Turing models. A very useful non-Turing approach to the complexity measurements is given by the *Information-Based Complexity* (IBC) approach. IBC has some very elaborated functional analysis tools. Given two infinite dimensional normed linear spaces F and G and a map $S : F \rightarrow G$, IBC studies how such a map might be best expressed as a finite dimensional objects function (relating the continuous real world with the discrete digital world).

Figure 3–2 illustrates the transition from the *problem space* F (continuous real world) to *analytic solution space* G (analytic world) using a continuous operator S . If the information operator \mathcal{N} is applied on an instance of the problem space then, it obtained a finite representation in \mathbb{R}^n of this problem. Now, it is possible to apply an algorithm operator ϕ on this finite representation and also possible to obtain an approximate solution.

In general, when it is applied \mathcal{N} to a $f \in F$ abstractly represent the extraction of information from f ; so, \mathcal{N} is called an information operator. The operator ϕ represents the computational procedure (algorithm) to process the information $\mathcal{N}\{f\}$ to approximate the solution $S\{f\}$. It is possible to define IBC how the mathematical paradigm that studies the problem of mapping of infinite dimensional spaces to finite dimensional spaces. The numerical cost (complexity) is assigned to algorithm ϕ . This work was interested in the smallest cost of ϕ for which

$$\sup_{\|f\| \leq 1} (\|S\{f\} - \phi\{\mathcal{N}\{f\}\}\| < \epsilon), \quad (3.1)$$

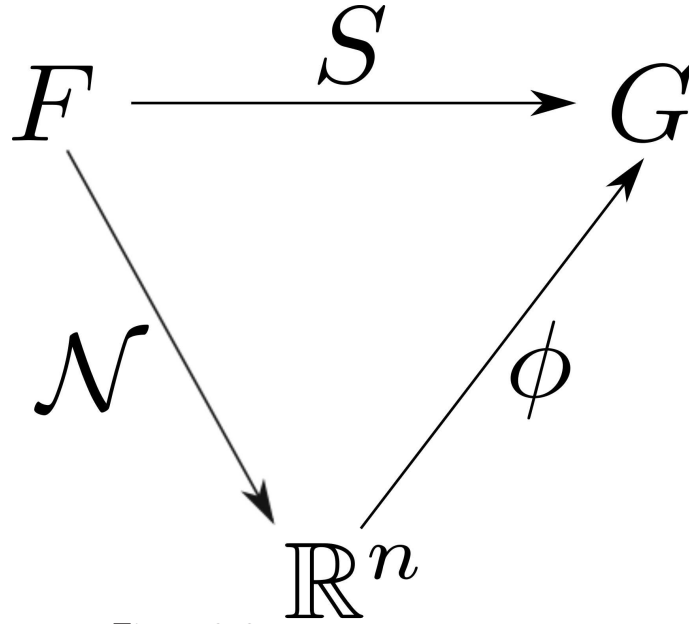


Figure 3–2: IBC Framework Spaces

for a given ϵ , where $\|\cdot\|$ is the norm in the space F and G . The focus is the minimization of the cost, based on the information extraction and error estimation.

This paper highlights the fact computers work with finite dimensional objects. Thus, the procedure followed in the computation would first involve a truncation in the instance of the *problem space* treated ($f \in F$) to something finite dimensional which can be represented in a computer. For example, f may be represented through its values at a finite set of points or through a finite number of coefficients in an eigenfunction expansion, or a finite number of Fourier coefficients. In this context, an algorithm ϕ is a mapping that transforms a truncated (discrete) problem into an approximate solution. The complexity of this transformation is measured using as reference a real number abstract machine. A numerical complexity can be assigned to the algorithm ϕ , based on the types and number of operations (e.g., additions, multiplications) the computation of ϕ performs.

This paper presents Information-based Complexity as a field of analytic (or continuous) complexity theory, as opposed to combinatorial complexity (classical or Turing complexity). In the classic complexity theory the cost of problems is

associated essentially with the number of permissible operations required to solve problems such as matrix inversion or multiplication. These combinatorial complexity problems are important; but, analytic complexity has consolidated as a distinct field. In the field of analytic complexity, the numerical bit operations do not play a central role, and basic arithmetic operations are treated as primitive operations. So, the approach in Information-Based Complexity is analytic rather than algebraic or combinatorial.

Relation to my thesis work: My thesis work focus the study of Information-Based Complexity as a tool to measure the computational complexity in Computational Signal Processing (CSP) frameworks. The use of this approach to analyze the complexity is justified by the fact of that my *problem space* (continuous analytic signals) must be truncated (quantized) and discretized previous to the processing stage on a digital computer. So, the signal processing algorithms work over truncated information and produce approximate solutions. In this scenario, this paper is important to contextualize the complexity analysis in my thesis work. The perspective in which the information is incomplete (truncated), noisy, and priced, describes very well the nature of my thesis research problem.

3.1.4 Improved RIP Analysis of Orthogonal Matching Pursuit

Matching Pursuit is a powerful heuristic to address compressive sensing problems [17]. The algorithm was designed as a statistical method for projecting multi-dimensional data onto interesting lower dimensional spaces. It was then introduced to the sparse approximation world in its non-orthogonal form Matching Pursuit by Mallat. This paper presents an improved Restricted Isometric Property (RIP) based performance guarantee to reconstruct T -sparse signals that asymptotically approach the conjectured lower bound given in Davenport, et al., [18].

This article extends the state-of-the-art by deriving reconstruction error bounds for the case of general non-sparse signals subjected to measurements of noise. The author then obtained generalized results for the case of K -fold Orthogonal Matching Pursuit.

The author presents an empirical analysis suggesting that *Orthogonal Matching Pursuit* (**OMP**) and K -fold **OMP** outperform other alternative compressive sensing algorithms in average case scenarios. This work concluded that these matching pursuit algorithms should perform roughly $T^{0.5}$ times worse than convex optimization and Iterative Thresholding algorithms.

OMP is a greedy alternative to convex optimization that solves the under-determined linear equation:

$$y = \Phi x, \tag{3.2}$$

where the vector x is a sparse signal, the matrix Φ is a short, fat measurement matrix, and the vector y is a small set of linear measurements of the signal. Only sparse approximation performance guarantees have been derived for **OMP**. These results depend on the coherence of the matrix Φ . The outcomes only bound the error $\|y - \tilde{y}\|_2$ where \tilde{y} is the estimated signal, produced by an **OMP** algorithm, which has a sparse representation in the column span of Φ .

The convex optimization algorithms tend to be slow; but, their solutions have very powerful error bounds based on a restricted isometric property. **OMP** works locally by attempting to select one non-zero entry of x in each iteration. A measurement matrix Φ satisfies a RIP of order T if there exists a constant $0 < \delta_T < 1$ such that $(1 - \delta_T \|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1 + \delta_T \|x\|_2^2))$ for all signals x that are T -sparse. The constant δ_T is called the restricted isometric number of order T .

The author clearly state the following important result: “An equivalent formulation of the RIP is that for every indexing set I of size T , here the $M \times T$ sub-matrix Φ_I generated by selecting the T columns of Φ corresponding to I , and satisfies: $1 - \delta_T \leq \text{Eigenvalues}(\Phi_I^* \Phi_I) \leq 1 + \delta_T$.”

K -fold Orthogonal Matching Pursuit. **KOMP** is almost identical to **OMP** except for the fact that K atoms are selected per iteration instead of 1. **KOMP** has two main advantages over **OMP**. The first one is speed: Given a T -sparse signal, one may use **KOMP** to recover the signal in T/K iterations versus the usual T iterations. Unfortunately, for accuracy, this method requires that all atoms selected per iteration be correct. The second one is mutually exclusive with the first, running T iterations of **KOMP** is selected a set S' of KT indexes where, with good probability, the signal's support set S is contained in S' .

Relation to my thesis work: In my thesis work three algorithms were used for parameter estimation in SISO and MIMO approaches. All of these algorithms are based on matching pursuit greedy algorithms. The first algorithm used was Basic Matching Pursuit, the second was Orthogonal Matching Pursuit, and the third was Least Square Recursive Matching Pursuit. The essence of these algorithms is to avoid direct computation in the cases where the inverse problem is associated with a sparse matrix. By exploiting this condition is possible to reduce the computational complexity derived of the algorithm. This paper is fundamental to gain understanding about the theoretical foundations associated with this type of algorithms.

4. IBC & CSP Theoretical Frameworks

4.1 Introduction

This chapter deals with the integration of fundamental concepts in Information-Based Complexity (IBC) and Computational Signal Processing (CSP). To the best of my knowledge, this integration approach has not been proposed in the scientific literature, up to this time. In stochastic systems, complexity measures such as Shannon information and Kolmogorov probability entropies have been widely used in communication and control systems; but, IBC has not been readily used as analysis tool.

4.2 Foundations of Computing

Computing may be defined as the processing of abstract entities. Processing is defined as a sequence of operations on a particular entity. An operation, in turn, is defined as a *functional action*; that is, an action with a purpose. Computer Science was developed in the past century and it reached a high formality level in the last decades. The list of pioneers in this branch include Alan Turing, Kurt Gödel, Emil Leon Post, Stephen Kleene, Alonzo Church, and John Hopcroft, among many others. Many of them were mathematicians, others were logicians, and others were electrical engineers. Their contributions allowed to give formalism to the nascent science of computing. Computer science studies the information processing potentialities of the theoretical systems called Finite State Automata (FSA). These are computational models endowed with a set of mathematical abstract structures for representing and processing information processes. Concepts such as *computability*, *unsolvability*, *complexity*, and others, are attached to FSA theory.

4.2.1 System or Machine Abstraction

Alan Turing is considered a precursor of the modern era of information. He presented in a formal manner an abstract computational model. It is now called the *Turing Machine* (TM). This theoretical model has been largely evaluated for determining the scope of its computational capabilities, having reached a high degree of acceptance. Many theoreticians consider the Turing Machine as the most powerful computational model[19] among the existing discrete computational models. In this work we adopt Sontag’s definition of a system or machine[20]. The formal Sontag’s system or machine M is as follows:

$$M = (\mathcal{T}, \mathcal{X}, \mathcal{U}, \phi), \quad (4.1)$$

where M is the system or machine, \mathcal{T} is the set denoting time, \mathcal{X} is the **state space** of M , \mathcal{U} is the **control-value** or **input-value space** of M , and ϕ is the **transition map** of M .

In 1936, Alan Turing and Alonzo Church postulated that any existing computational model had the same algorithmic capabilities, or a subset of the capabilities of the standard Turing Machine. This assertion is known as the *Church-Turing Thesis*. Since then, many computational models have been proposed; however, it has been always possible to show that their computational power was equivalent to the Turing Machine computational power.

Many principles of modern computational theory were developed under the Turing Machine perspective, so basic issues as *computational complexity* have been studied using the Turing Machine model for evaluating and testing of algorithms. Under this assumption, all computational resources in most digital computers are quantified using the Turing Machine model as the standard reference[21]. A consequence of this approach is a *complexity theory* based on the Turing Machine model [22]. This approach at addressing the *complexity issues* has been called *classical complexity*

theory approach [23]. Also, it is a common belief that the *classical approach* is the unique manner to address a large class of complexity problems.

Computational complexity is priorly evaluated using two main resources: time and space. The *classical complexity theory* has categorized many computational problems, based on a non-deterministic Turing Machine model, using a well known taxonomy: deterministic polynomial time problems (\mathbf{P} set), non-deterministic polynomial time problems (\mathbf{NP} set), non-deterministic polynomial complete (\mathbf{NP} -complete set), and non-deterministic polynomial hard (\mathbf{NP} -hard set). Figure 4–4 shows the \mathbf{NP} taxonomy in case \mathbf{NP} set is assumed to be not equal to the \mathbf{P} set (this assertion is only a conjecture).

The Turing Machine has been modified for trying to improve its computational power. The original Turing Machine has an infinite tape, a read/write head, and this head can read or write a single character at a time. Each change of state must be a consequence of a mapping recorded in the transition's function. Some variants of the original Turing Machine include a two-tape machine, multi-tape machine, multi-dimensional tape machines, among others. The theoreticians have shown that all Turing Machine variants have the same computational power of the original Turing Machine; i.e., these variants can solve the same set of problems that the original Turing Machine it is able to solve. This fact validates the Church-Turing thesis. So, the *classical theory of computation* considers *computable* any problem that can be solved using a Turing Machine.

Beyond the problem of *computability* the variants of the Turing Machine have shown another important issue: the computational complexity of a problem depends on the computational machine used. The number of steps to resolve a problem is different if we use an original Turing Machine instead of a multi-tape Turing Machine. However, the issue can be more complex. If we decide to establish other resources (different, say, to time) to measure complexity; e.g., energy or space used

to resolve the problem; then, complexity measurements can depend on multi-variable elements.

In general, complexity theory studies natural and artificial problems. It classifies these problems according to defined taxonomies, measuring the computational efficiency of various algorithms and computational paradigms. Finally, complexity theory establishes comparative rules between the complexity measurements in each case.

4.2.2 Complexity Classes

Complexity theory analyzes the most efficient known algorithms to resolve a problem. Using the complexity measurement associated with an algorithm each problem is classified into a *complexity class*. If the resource measured was *time*, then the problem is classified into **DTIME** class, or **NTIME** class. The classification depends on the type of Turing Machine required to reach the solution: *Deterministic Turing Machines (DTM)* or *Non-deterministic Turing Machines (NTM)*. The *time complexity* of an algorithm is determined by the number of steps employed for reaching a solution as a function of the cardinality of the input data set. The *space complexity* of an algorithm is the amount of memory used by it for reaching a solution as a function of input data set cardinality. Analogously, a problem can be classified into a *space complexity class* depending on the amount of memory used by it for reaching a solution. The **DSPACE** and **NSPACE** classes are related to deterministic and non-deterministic Turing Machines.

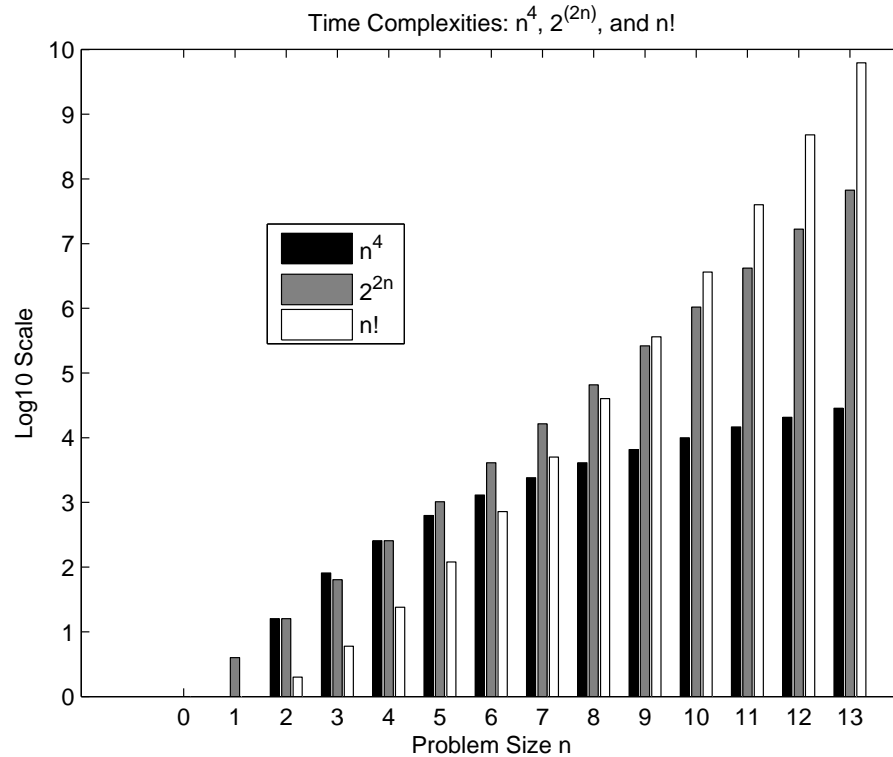


Figure 4–1: Number of operations in n^4 , 2^{2n} , and $n!$ complexities.

4.2.3 Finite-State Automaton

To model hardware systems it is required to introduce the notion of automaton (*automata* in plural). An *automaton* is an abstraction that possesses all the indispensable features of a digital computer system. This automaton receives inputs, produces outputs, may have memory storage, and makes decisions during the transformation process from input to output.

A *formal language* is an abstraction of the general characteristics of programming languages. A formal language consists of a finite set of *symbols*, called an alphabet, and some rules of formation by which these symbols can be combined in entities called *strings*. A string is a finite sequence of symbols. A formal language is the set of strings generated using the formation rules. The basic computational model, to process strings and decide whether they belong to a formal language, is the *finite state automaton* (FSA). Some theoretical machines more complex than FSA

can be found in the literature. We will use these theoretical machines to formulate computational structures, and simulation environments like Field Programmable Gate Arrays (FPGA) and other hardware simulation devices.

Formally, an *automaton* is a system D which recognizes strings $s \in \Sigma^*$, where Σ is an alphabet, and Σ^* , called *Kleene Star* of Σ , is defined as the set of all strings forming finite sequences whose symbols, come from Σ . The set Σ^* include the empty or null string λ . The strings “accepted” by D form a set $L \subseteq \Sigma^*$ and it become a formal language. Thus, each automaton D recognizes a language contained in Σ^* . This language is denoted $D(L)$. The theoreticians are very interested in the study about languages recognized by automata. The automata can be categorized as *deterministic* and *non-deterministic*. A deterministic automaton is formally denoted by a quintuple $D = (\Sigma, Q, q_0, F, A)$, where Q is a finite set of states, $q_0 \in Q$ is the start state, $F : Q \times \Sigma \rightarrow Q$ is a transition function, and $A \subseteq Q$ is a set of accept states. The dynamic in this kind of machines is simple. The automaton D receives a input string $s \in \Sigma^*$ conformed by a finite sequence of elements of Σ , then starting in state q_0 each element s_i is read and the transition function F indicates a change of state. If the last current state in D is $q_j \in A$, then D accepts; however, if the last current state in D is $q_k \notin A$ then D rejects.

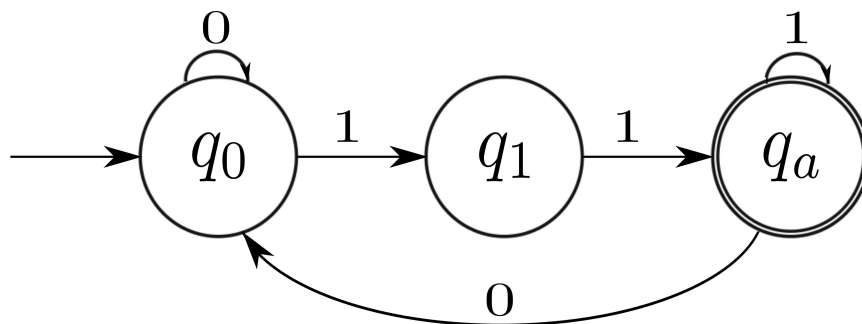


Figure 4-2: Graphical Representation of a Finite State Automaton

Figure 4–2 shows a graphical representation of a finite state automaton (FSA) that recognizes a formal languages on $\{0, 1\}^*$ whose strings end with the sequence “11”.

4.2.4 Formal Definition of a Turing Machine

A Turing Machine can be defined in a formal manner as a quadruple

$$M = (Q, \Sigma, \delta, q_0), \quad (4.2)$$

where Q is a set of states. This set must be *finite* and non-empty, i.e., any *feasible* Turing Machine must have a finite number of states for a realizable implementation. Among the states of Q we can find two *special* states: a *accept* state (q_a , first state in Q) and a *reject* state (q_r , second state in Q). $q_0 \in Q$ and is called *initial* state. The initial state can be any state in Q , so the minimum cardinality of Q must be 2. Σ is the *alphabet* of the machine. Any alphabet is a finite set of symbols. The alphabet Σ must have at least a *special* symbol called *blank* symbol (\sqcup) and other symbol to constitute strings. The sets Q and Σ must be disjoint sets. δ is a finite collection of *transition rules* called *transition function*. Each *transition rule* maps a tuple in $(Q \times \Sigma)$ to a tuple in $(Q \times \Sigma \times \{LEFT, RIGHT\})$. The transition function δ can be considered the logical core of the Turing Machine. Each transition must be read in the following manner: in a time t the Turing Machine M is in a state $q_m \in Q$ and its read/write head reads from the tape a symbol $s_k \in \Sigma$; then, M changes to a state $q_n \in Q$, the read/write head writes to the tape a symbol $s_l \in \Sigma$, and, finally, the head is moved one position to the RIGHT or LEFT over the tape. Some variants of Turing Machines accept that the head stays in the same position on the transition. But, this feature don't add more computational power to the model. Other Turing Machine variants add a new set Γ as *alphabet tape*; however, only one alphabet set can represent, in a successful manner, the Turing Machine alphabet. Each *instruction* is represented by a quintuple (q_m, s_k, q_n, s_l, A) . Where A represents

the direction of head's movement (RIGHT or LEFT). A Turing Machine receives as input a string $s \in \Sigma^*$ written over the tape and eventually halts in an accept or reject state. The transition function, as previously defined, allows only, in each step, a change of a state q_m to a state q_n . This transition function categorizes a very important type of Turing Machine called *deterministic Turing Machine* (**DTM**). All problems that can be resolved in *polynomial time* as a function of the *input string length* using a deterministic Turing Machine are classified in a *time complexity class* called **P**.

No all Turing Machines are deterministic Turing Machines. If the transition function δ has in its co-domain elements of the powerset of Q instead of single elements of Q , then we can say that δ is a non-deterministic transition function and therefore the associated Turing Machine is called a *non-deterministic Turing Machine* (**NTM**). Formally, N is a non-deterministic Turing Machine if its transition function δ performs a mapping from a tuple in $(Q \times \{\Sigma\} \cup \lambda)$ to a tuple in $(\mathcal{P}(Q) \times \Sigma \times \{RIGHT, LEFT\})$. Where λ is the null string ($\lambda \in \Sigma^*$) and $\mathcal{P}(Q)$ is the powerset of Q . The main consequence of λ -transitions is the occurrence of spontaneous transitions; i.e., transitions triggered without any symbol read. Figure 4-3 illustrates a non-deterministic computation tree.

All problems that can be resolved in *polynomial time* using a non-deterministic Turing Machine belong to *time complexity class* called **NP**. Another alternative definition says that all problems whose solutions can be *verified* in polynomial time belong to the **NP** class. Although any non-deterministic Turing Machine has an equivalent deterministic Turing Machine, the procedure to perform the conversion is not executed in polynomial time. So far it is clear that $\mathbf{P} \subseteq \mathbf{NP}$. But one of the seven *millennium prize problems* proposed by the Clay Mathematics Institute in 2000 addresses the question if whether or not all problems in **NP** are also in **P**. It is still an unanswered question. The **NP**-Hard class is the set of all decision problems

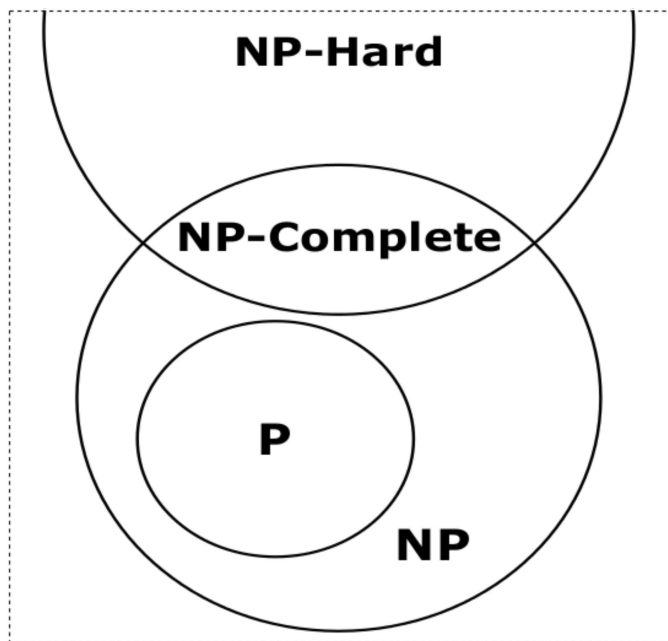


Figure 4–4: Complexity Classes: Polynomial (P), Non-Deterministic Polynomial (NP), NP-Complete, and NP-Hard

Big O Notation	Type of Complexity
$O(1)$	Constant
$O(\log(n))$	Logarithmic
$O((\log(n))^c)$	Polylogarithmic
$O(n)$	Linear
$O(n^2)$	Quadratic
$O(n^c)$	Polynomial
$O(c^n)$	Exponential
$O(n!)$	Factorial

Table 4–1: “Big O” Notation Associated to Classical Complexity Class

of the input data set, but $O(N^2)$ represents an algorithm whose time execution is directly proportional to the square of the size of the input data set.

Big O notation (also called *Landau’s symbol*) describes the asymptotic behavior of a function when this function represents the amount of resources (time or space) used by an algorithm to solve a given problem. Table (4–1) categorizes some algorithms by their asymptotic behavior.

4.4 Information-Based Complexity (IBC) Definition

We begin with the definition of IBC: “It is the study of the inherent difficulty in solving problems for which only partial, noisy, and costly information is available. Since information is partial and/or noisy, it causes uncertainty in the solution.” (see [24]).

Continuous complexity theory gets its name from the models of mathematical computation on which it is based. Modeling works inside a digital computer takes center stage in complexity theory, and the complexity of problems is measured in terms of bit operations using the Turing machine model. In IBC mathematical analysis is used as the primary tool for modeling works instead of the combinatorial techniques utilized in classical computational complexity.

In the continuous complexity model (or real-number model), real number operations such as multiplication and function evaluation are defined as functional primitives, and complexity analysis has a more analytic sense. The real number model of computation is the basis for recent foundational work in computation theory, and for older work on zero finding for polynomials and other equations. The notion of partial information is an important element in the theory of continuous computational complexity. Computations in numerical analysis attempt to model infinite dimensional objects, while computers, deal with finite dimensions [25]. IBC mainly focuses on infinite-dimensional problems, but there has been some work on finite-dimensional and discrete problems.

Computational and numerical analysis problems can be reduced to computing an output from an input. Define S to be the mapping from input to desired output. For example, the input may consist of a function f on the unit interval. Information $N(f)$ about f might be values of f at a discrete set of points, a common information set in numerical analysis. Desired output $S(f)$ might range from the integral $f =$

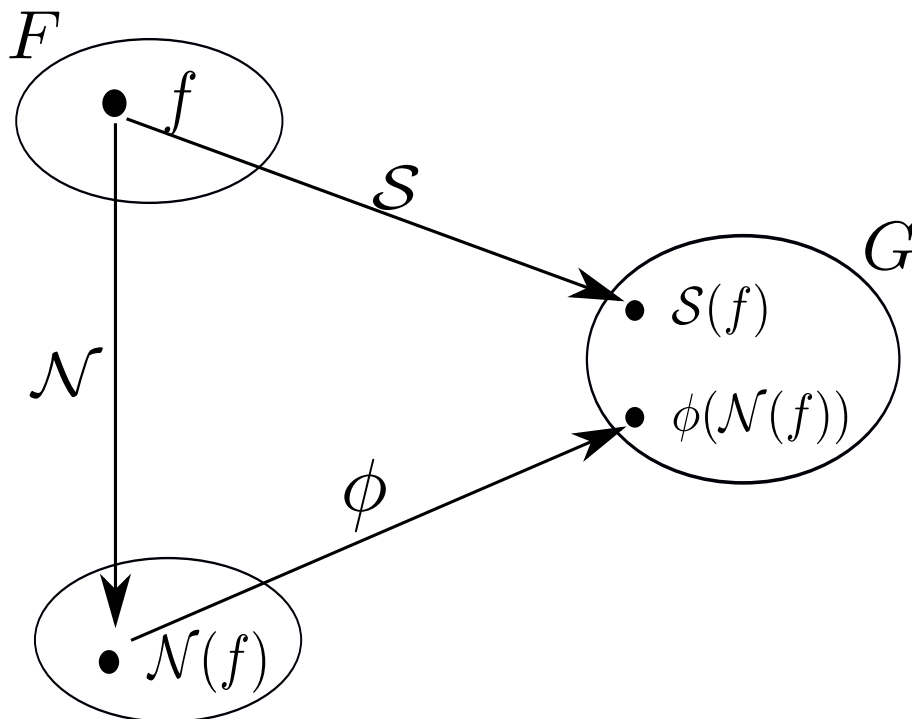


Figure 4–5: Traub-Werschulz IBC Framework

$\int_0^1 f(x)dx$, to recovery of f itself (from partial information $N(f)$). The goal is then to compute a very good approximation, such that the error is minimized.

The notion of partial information is central in computational solutions of mathematical problems. Indeed there are only two ways to get one's hands on the functions and operators in numerical problems for which one seeks computational solutions. The first is to describe them analytically or symbolically, manipulate them in exact form until a solution is obtained, and finally extract the finite collection of numbers we need to work with. The second and more common way is to start with information $N(f)$ about inputs in partial form as numbers we must do something with. Thus N has filtered f into the finite dimensional form $N(f)$. Figure 4–11 shows this process [25]. Is there a better approach? We asked ourselves this question during our research endeavor.

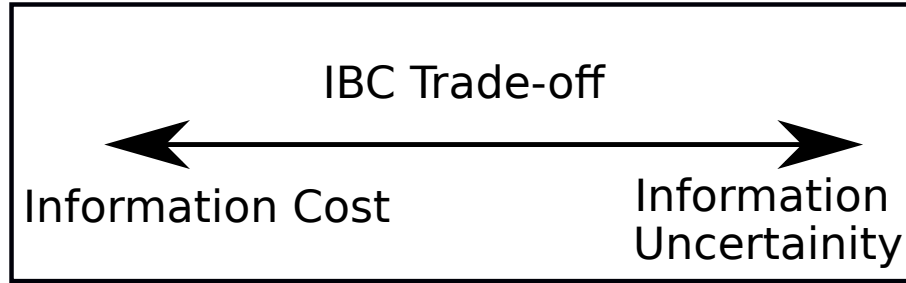


Figure 4–6: IBC Trade-Off

Information-based complexity (IBC) is the theory that studies all subjects relative to complexity, based on real-numeric paradigms. IBC presents a trade-off that is illustrated in Figure 4–6.

In general, four worlds are represented in IBC theory. Figure 4–9 is a depiction of these four worlds as described by J.F. Traub and A.G. Werschulz in the monograph entitled “Complexity and Information” and published in 1998 by Oxford University Press [26]. We have adopted this four-world representation as a setting for our proposed Computational Signal Processing modeling framework. However, we made some slight changes to this representation in search for a better fit to our conceptual framework. Our reformulation of Traub-Werschulz’s four-world representation is depicted in figure 4–10. In our depiction we want to strongly emphasize the natural versus the virtual worlds as well as the continuous versus the discrete computational models. The complete process is illustrated in the Figure 4–7.

In this thesis we define a mathematical model as a set of mathematical expressions which describe an aspect of the physical world. A main property of this set is that it allows to make inferences about attributes of the aspect of the physical world being described.

4.5 Assumptions of IBC

This thesis assumes that events are carried out in the physical world (or natural world). These events may be modeled using physical laws and mathematical tools.

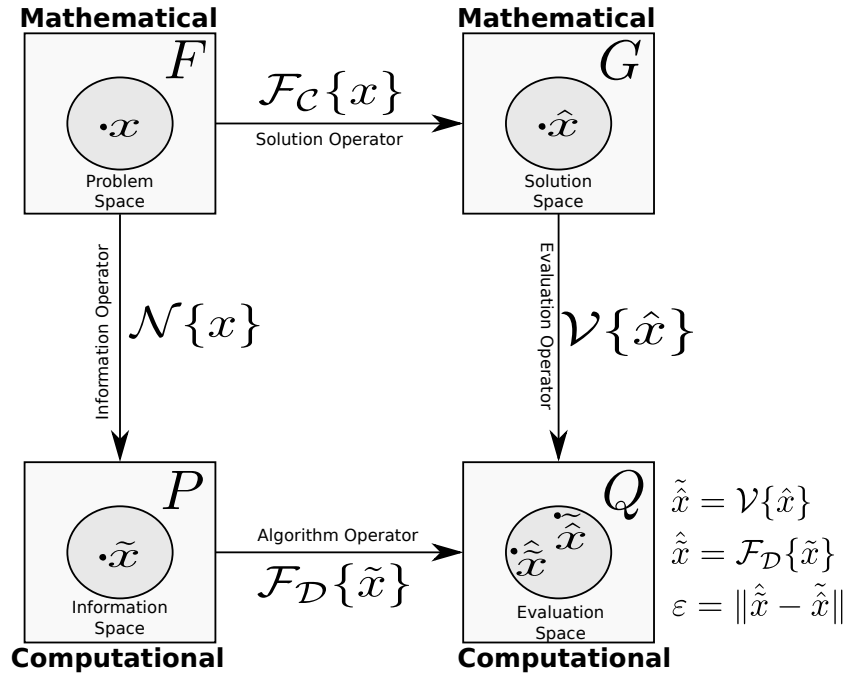


Figure 4–7: Complete Transformation Process using Two Different Approaches

These models are mostly continuous models, expressed using equations and general mathematical expressions. A computer simulation can be performed for verifying the accuracy of the model. This simulation must be supported by a model of appropriate computation. IBC has three principal assumptions about of information,

- the information is partial,
- the information is tainted, and
- the information is priced.

We deal with partial information when solving an infinite-dimensional problem using a Turing computers. The reason is that the Turing computer can handle only finite sets of numbers and finite sets of states. Infinite objects, such as real-value functions, are replaced by mappings with finite co-domains. So, the values of the functions are computed using round off techniques. Thus, the information is now corrupted by round-off errors, measurement errors, or any other kind of errors induced by the sampling and quantization processes. In this sense, the primary

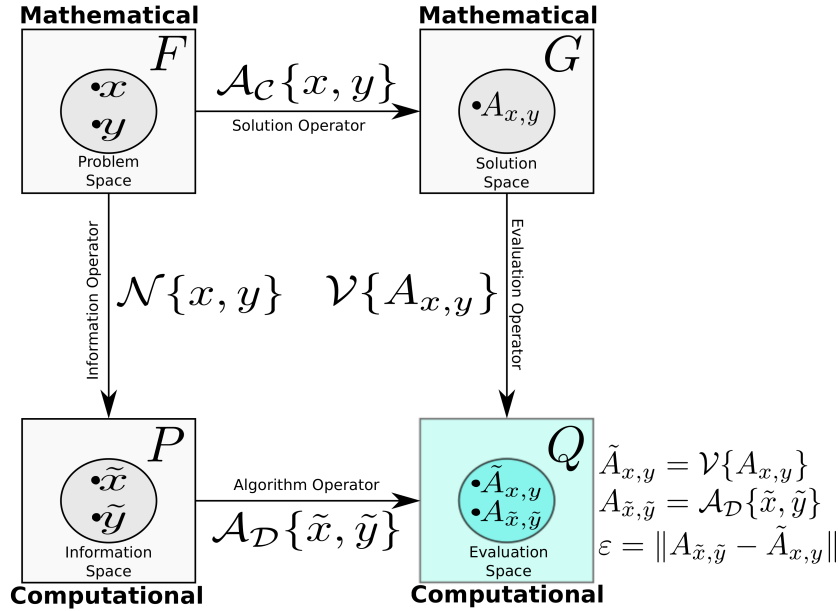


Figure 4–8: IBC Spaces. Ambiguity Function Case

Real-World Phenomena	Computer Simulation
Mathematical Model	Model of Computation

Figure 4–9: Four Worlds Described by IBC Theory

cost of solving any problem centers on computing the functions values, and the information is considered priced, expensive (computationally).

To motivate these assumptions about information, consider a continuous dynamical system. It consists of an equation that determines how the system evolves over time and an initial condition. Assume that the equation depends on coefficients that are real functions of a real variable. Since a Turing computer can store only a finite set of numbers, these functions must be replaced by such finite sets. Therefore, we have only partial information about the equation of evolution and, similarly, about the initial conditions.

Physical World Phenomena (Natural)	Digital World Abstraction (Virtual)
Mathematical Model (Continuous)	Computational Model (Discrete)

Figure 4–10: A Reformulation of the Four Worlds Described in IBC Theory

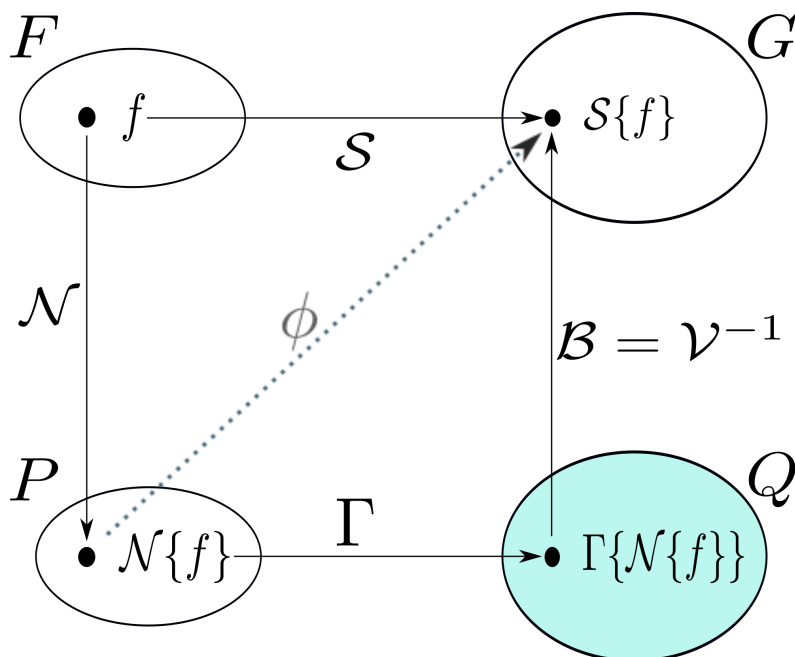


Figure 4–11: Modified Traub-Werschulz IBC Framework

4.6 Application Fields for IBC

The applications of IBC typically involves multivariate problems of science and engineering. Examples include high dimensional integration, ordinary and partial differential equations, nonlinear optimization, function approximation, and ill-posed problems. Although the focus has been on the complexity of computations, the complexity of verification and implementation testing has also been studied.

In this section we present a very simple but illustrative example on the use of IBC to assist in the selection of an algorithm to obtain an approximate solution to the problem of computing the continuous-time Fourier transform of a finite energy signal. We show how to obtain a solution using mathematical analysis; that is, we provide a solution by analytical means. We also provide an approximate solution by first obtaining partial information about the prescribed problem element and the proceed to use a **P**-class algorithm to arrive at the solution.

Let $L_p(\mathbb{R})$ represent the set of all complex-valued functions, say f , over the real numbers \mathbb{R} such that, for $1 \leq p \leq \infty$, we have

$$\|f\|_p \triangleq \int_{t \in \mathbb{R}} |f(t)|^p < \infty, \quad (4.3)$$

where the integral expression is defined as a Lebesgue integral. For discrete signals or sequences, say s , $l_p(\mathbb{Z})$ represents the set of all sequences such that, for $1 \leq p \leq \infty$, we have

$$\|s\|_p \triangleq \sum_{n \in \mathbb{Z}} |s[n]|^p < \infty, \quad (4.4)$$

where the sum is taken over the integers \mathbb{Z} .

The set $L_\infty(\mathbb{R})$ is conformed by all complex-valued functions, say f , over the reals \mathbb{R} , such that

$$\|f\|_\infty \triangleq \underbrace{\text{ess sup}}_{t \in \mathbb{R}} |f(t)| < \infty. \quad (4.5)$$

Let the set $l_\infty(\mathbb{Z})$ represent the signal space of all sequences, say s , over the integers \mathbb{Z} , such that

$$\|s\|_\infty \triangleq \max|s| < \infty. \quad (4.6)$$

IBC Example: A good example, on the time-frequency analysis, is the computation of the Fourier transform of a continuous signal. Example: Computing Fourier transform of $h(t) = 2e^{-5t}\mu(t)$.

$$H(f) = \int_{-\infty}^{\infty} 2e^{-5t}\mu(t)e^{-j2\pi ft} dt = 2 \int_0^{\infty} e^{-5t}e^{-j2\pi ft} dt, \quad (4.7)$$

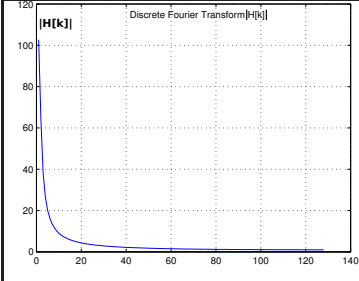
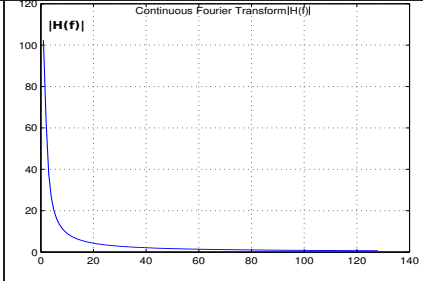
$$H(f) = \frac{2}{j2\pi f + 5}. \quad (4.8)$$

Table 4-2: Matlab Computation for Two Approaches (Discrete and Real-Number)

$N = 256;$	number of samples
$T = 1/N;$	sampling period
$t = 0 : T : 1 - T;$	support region for $h[t]$
$h_t = 2 * \exp(-5 * t);$	$h[t] = 2e^{-5t}$
$H_k = fft(h_t);$	$H[k] = \sum_{n=0}^{N-1} 2e^{-5t}e^{-j2\pi kn}$
$H_k = H_k(1 : end/2);$	symmetric spectrum
$f = 0 : N/2 - 1;$	support region for $H(f)$
$H_f = N * 2 ./ (j * 2 * \pi * f + 5);$	$H(f) = N * \frac{2}{j2\pi f + 5}$

This example shows the computation of Fourier transform of the function $h(t) = 2e^{-5t}\mu(t)$. The Equation 4.8 represents the Fourier transform, $H(f)$, of $h(t)$ in analytical form. The Table 4-2 presents documented Matlab pseudo-code for both approaches (discrete and analytic real-number computation), for the same transform. H_k is computed in discrete form (using a discrete h_t), but H_f is computed using analytical resources. The results are compared in the Table 4-3.

Table 4–3: Comparative Table of Results of H_k and H_f Transforms

	
$H_k(1 : 5)$	$H_f(1 : 5)$
102.40	102.71
39.70 - j49.89	40.43 - j49.55
14.00 - j35.17	14.90 - j34.93
6.73 - j25.38	7.68 - j25.19
3.90 - j19.60	4.87 - j19.45

4.7 Works on IBC Field

A number of works have been written on IBC. The monograph entitled Complexity and Information [25], of J.F. Traub, was published in 1998. Other works include:

- Sikorski, K., Optimal Solution of Nonlinear Equations [27], Oxford University Press, Oxford, UK, 1998.
- Keller, A., Quasi-Monte Carlo Methods for Photorealistic Image Synthesis [28], Shaker, Aachen, 1998.
- Frank, K., Optimal numerical solution of multivariate integral equations [29], Shaker, Aachen, 1997.
- Plaskota, L., Noisy Information and Computational Complexity [30], Cambridge University Press, Cambridge, UK, 1996.

- Werschulz, A. G., The Computational Complexity of Differential and Integral Equations: An Information-Based Approach [31], Oxford University Press, New York, 1991.
- Novak, E., Deterministic and Stochastic Error Bound in Numerical Analysis [32], Lecture Notes in Mathematics, vol. 1349, Springer-Verlag, New York, 1988.

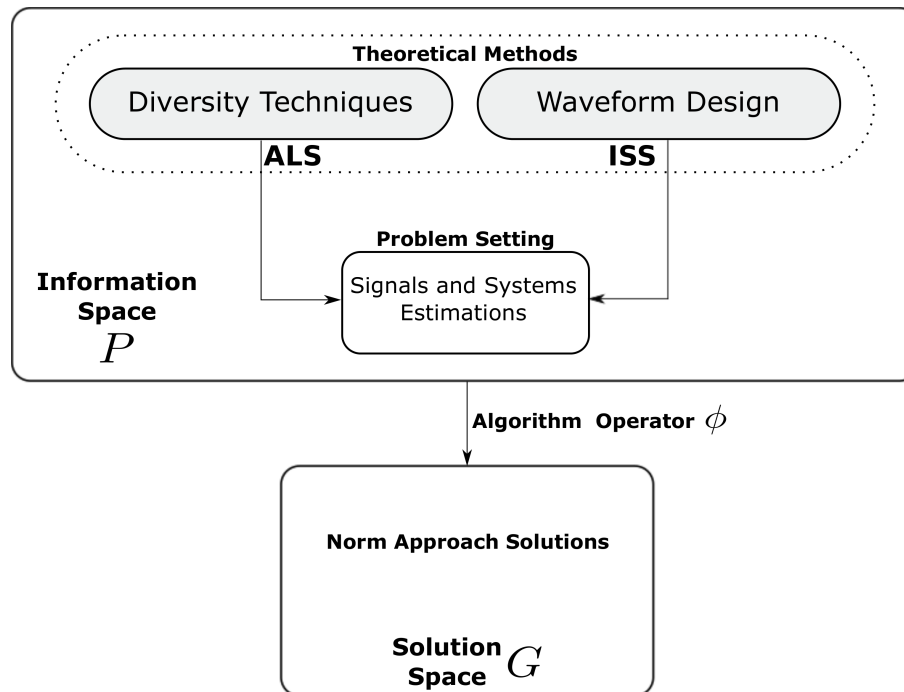


Figure 4–12: Information Space to Solution Space Transformation

4.8 Computational Signal Processing Framework

The Introduction chapter of this thesis presented the concept of Computational Signal Processing as a branch of Computational Complexity, the discipline which studies the intrinsic difficulty of mathematically-posed problems and seeks optimal means for their solution. In a more general sense, Computational Signal Processing (CSP) deals with the treatment of physical signals using computational methods. We reiterate that a computational method is defined as a non-empty structured set of computational tools utilized to solve mathematically posed problems.

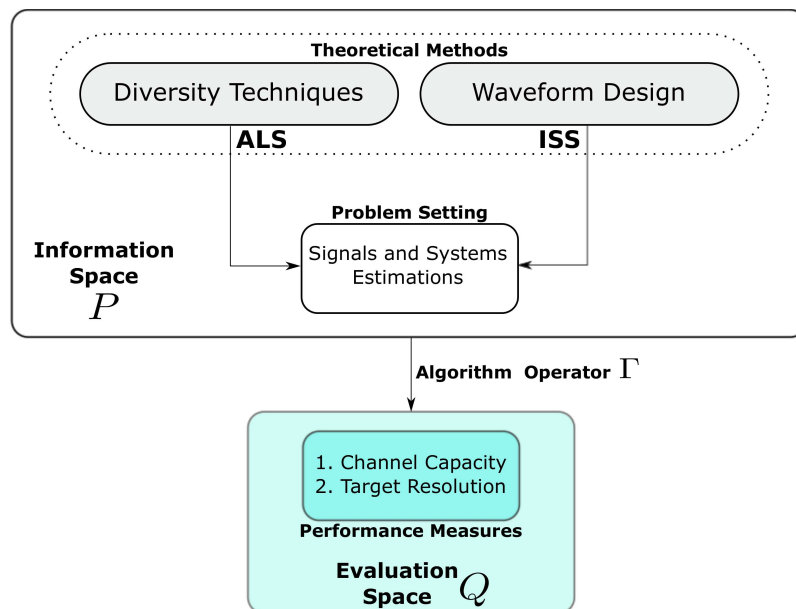


Figure 4–13: Information Space to Evaluation Space Transformation

Extended concepts and methods relating fundamentals of Computational Signal Processing with Information-Based Complexity were formulated by Domingo Rodríguez [33].

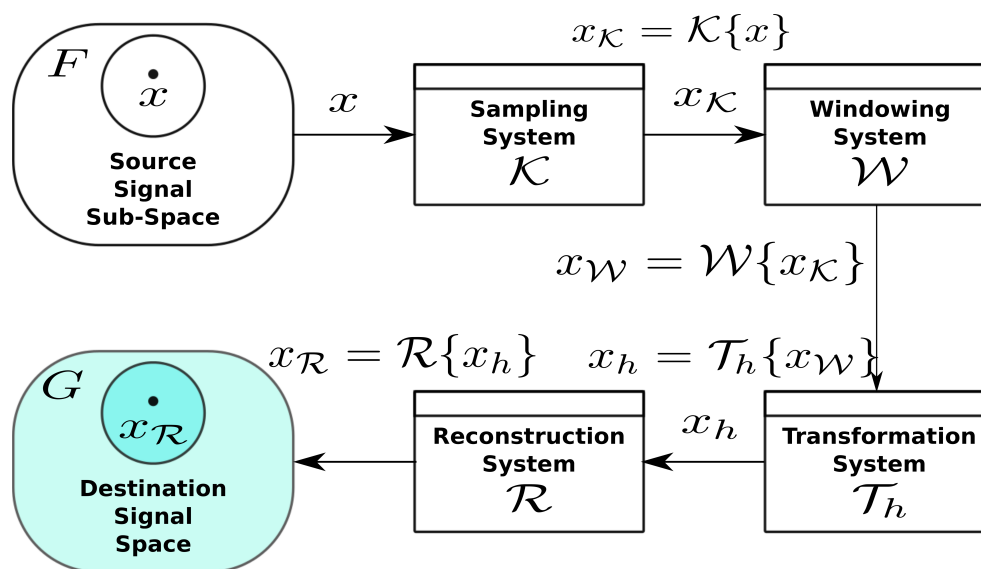


Figure 4–14: Computational Signal Processing (CSP) Framework

Physical signals are manually modeled as stochastic processes. A physical signal serves as the entity which carries the information about the state of a physical system or phenomenon. In this context, a physical signal describes a physical quantity, usually as a function of time or any other independent variable. Thus, a physical quantity is a physical property of a physical system or phenomenon that can be quantified by measurement. Any entity is measurable if it can be expressed, through an *experimental estimation process*, as a ratio of a prescribed unit of measurement. The device performing the experimental estimation process may be defined as a signal processor which interacts with the physical system transforming the information emanating from such system. For example, given an underwater communication medium as a physical system, an acoustic transducer may serve as a signal processor which converts a physical quantity, in this case a pressure field signal, into an stochastic voltage or current signal. It is advantageous to treat signals as belonging to an ensemble, set, or family, with common attributes. Ensembles or sets of signals with particular attributes are usually called signal spaces. In this research work I studied band-limited acoustic signal sets as special linear signal spaces.

In this thesis work I also concentrated on the development of underwater communication channel models to integrate with underwater point-target scattering models in order to construct a computational modeling framework capable of characterizing underwater moving objects modeled as point-targets. Franz Hlawatsch and Gerard Matz have demonstrated that a Rayleigh fading channel model is appropriate for describing attributes of a communication channel assumed to be wide-sense stationary (WSS), with uncorrelated scattering (US) [34]. The channel is also assumed to be underspread; that is, the product of the average delay spread τ by the average Doppler spread ν is much less than 1, i.e. $\tau\nu \ll 1$.

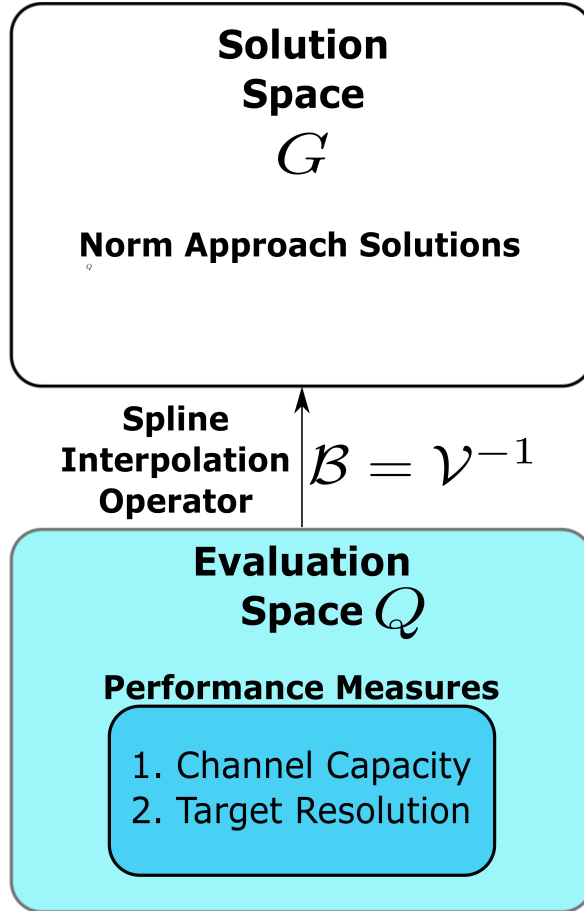


Figure 4–15: Evaluation Space to Solution Space Transformation

These type of channels are usually analyzed under an information theoretic setting. Thus, the underlying channel models are usually interpreted as information-based models. Under a CSP framework, these models are interpreted as information-based computational models. This important realization allowed us to naturally introduce principles of information-based complexity (IBC) to further enhance these communication channels models.

In this work, Figures 4–12, 4–13, and 4–15 represent an original contribution in the field of IBC theory. The representation of a new “*Evaluation Space*” allows to establish a relationship between IBC theory, approximation algorithms, and optimization theory. These branches have been used in independent manner in signal

processing theory, but now, we present a unified framework to address signal processing problems like channel parameter estimation. This problem has a very high complexity, but we propose a approximate solution which minimizes the error to acceptable levels.

This thesis work established a relation between the *Traub-Werschulz (TW) IBC Framework* and the *Computational Signal Processing* approach developed during this research. Figure 4–14 illustrates the CPS Framework, where the spaces F and Q denote the same spaces in the TW IBC Framework.

4.9 Convex Optimization

This section establishes the foundations for parameter optimization in the channel estimation problem.

4.9.1 General Optimization Problem

A *general optimization problem*, has the following form

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq b_i, \quad i = 1, \dots, m, \end{aligned} \tag{4.9}$$

where the vector $x = (x_1, \dots, x_n)$, is called the *optimization variable*, the function $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$, is called the *objective function*, the functions $f_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, m$, are called *constraint functions*, and the constants b_1, \dots, b_m are the bounds for the constraints.

Let x^* be a vector in \mathbb{R}^n , it is a *solution* of the optimization problem if it has the smallest objective value among all vector that satisfy the constrains. So, for any w with $f_1(w) \leq b_1, \dots, f_m(w) \leq b_m$, we know that $f_0(w) \geq f_0(x^*)$.

4.9.2 Particular Optimization Problems

If we impose some additional constraints to the general optimization problem we can obtain some very interesting particular optimization problems. In the following sections we present some of them.

4.9.3 Linear Programming Problem

If the objective function f_0 and the constraints functions f_1, \dots, f_m are linear, satisfying

$$f_i(\alpha x + \beta y) = \alpha f_i(x) + \beta f_i(y), \quad (4.10)$$

for $x, y \in \mathbb{R}^n$, and $\alpha, \beta \in \mathbb{R}$.

If the referred functions don't comply with the linear property then the optimization problem is called *nonlinear program*.

4.9.4 Convex Optimization Problem

A *convex optimization problem* is one which the objective function f_0 , and the constraint functions f_i are convex, i.e., they satisfy the additional condition

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y), \quad (4.11)$$

for $x, y \in \mathbb{R}^n$, and $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$.

If we replace the inequality in the Equation (4.11) with an equality, then we obtain the Equation (4.10). So, the convex optimization is a generalization of the linear programming problems.

4.9.5 Least-Square Problem

A *least-square problem* is an optimization problem without constraints ($m = 0$), and an objective which is a sum of squares of terms of the form $a_i^T x - b_i$:

$$\text{minimize } f_0 = \|Ax - b\|_2^2 = \sum_{i=1}^k (a_i^T x - b_i)^2, \quad (4.12)$$

where $A \in \mathbb{R}^{k \times n}$, $k \geq n$, a_i^T are the rows of A , and the vector $x \in \mathbb{R}^n$ is the optimization variable.

The solution of the problem is attained resolving the following linear equations system,

$$(A^T A) x = A^T b, \quad (4.13)$$

where $x = (A^T A)^{-1} A^T b$.

Sometimes it is possible to solve the least-square problems by exploiting some special structures in the matrix A , for example its sparsity. This special condition will be essential in the channel parameter estimation problem. The least-square problem can be resolved in $\mathbf{O}(n^2k)$, where n is dimension of the vector x and k is a constant such as $k \geq n$. We usually solve the least-square problem much faster than $\mathbf{O}(n^2k)$ exploiting the sparsity condition. The least-square problem is fundamental in regression analysis, and other parameter estimation and data fitting methods. Its statistical interpretation is associated with maximum likelihood estimation of a vector x , given linear measurements affected by a noise signal.

4.9.6 Approximation Algorithms

Many significant problems in computer sciences are so difficult to solve in an optimal manner. Some of these problems belong to complexity class **NP-Hard**. In some cases polynomial-time algorithms do not exist in order to reach an optimal solution. Two examples can be the *euclidean traveling salesman* (ETS) problem and the *independent set* problem. The first problem is defined as: given a set of points on the plane, find the shortest path that visits all the points. The second problem is defined as: given a graph $G = (V, E)$, find a maximum-size independent set $V^* \subset V$. A subset is called independent if no two vertices in the subset are connected by an edge.

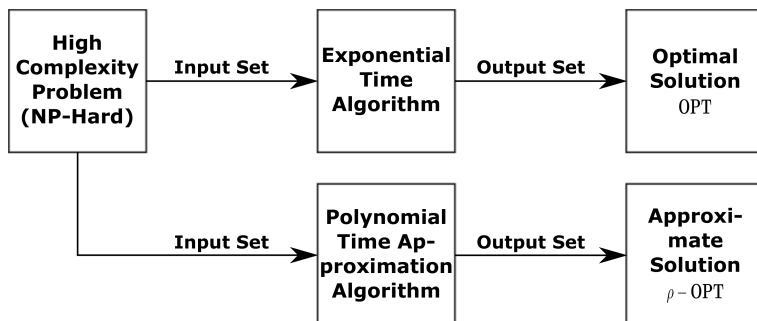


Figure 4-16: Approximation Algorithm Approach

How to address this type of problems if we have not polynomial-time algorithms available? Only if the cardinality of the input set is very small, an exponential-time algorithm is useful. We can relax the *optimal solution* condition by *near to optimal solution* guarantee.

In the case of the ETS problem, if the *optimal solution*, **OPT**, is computationally very expensive, then, we can use an *approximation algorithm* ϕ that provides a guarantee of $\phi(S) \geq \rho \times \mathbf{OPT}$, that is, a solution ρ -approximate to **OPT**. We call an algorithm producing a solution that is guaranteed to be within some factor of the optimum an approximation algorithm. Other approaches, like heuristic approaches, can produce optimal solutions in some specific cases, but can not offer guarantee of approximation to optimal solutions.

Any algorithm ϕ for a minimization problem is called a ρ -approximation algorithm, with $\rho > 1$, if the algorithm $\phi(S)$ produces, for any input set S , a solution whose value is at most $\rho \times OPT(S)$. But, any algorithm ϕ for a maximization problem is called a ρ -approximation algorithm, with $\rho < 1$, if the algorithm $\phi(S)$ produces for any input set S a solution whose value is at least $\rho \times OPT(S)$. Figure (4-16) shows the relation between an optimal algorithm and an approximated algorithm. The factor ρ is called the approximation factor.

Three approaches for addressing **NP-Hard** problems can be:

- Brute-force algorithms.
 - Develop clever enumeration strategies.

- Guaranteed to find optimal solution.
- No guarantees on running time.
- Heuristics.
 - Develop intuitive algorithms.
 - Guaranteed to run in polynomial time.
 - No guarantees on quality of solution.
- Approximation algorithms.
 - Guaranteed to run in polynomial time.
 - Guaranteed to find *high quality* solutions.
 - Weakness: the need to prove a solution value is close to optimum, without even knowing what optimum value is.

Among these options the *approximation algorithms* is a good choice in parameter signal estimation problems.

5. Acoustic Linear Stochastic Systems

5.1 Introduction

The study of underwater acoustic communications has been growing in recent decades. The marked differences between the acoustic waves and electromagnetic waves have allowed the development of a research field apart of traditional communications. The characterization of underwater acoustic channels has become a very important research subject. This problem is particularly interesting when the transmission medium for the signals is shallow water.

Communication channels are modeled as linear time-variant (LTV) systems [35]. Based on this idea, many models have been proposed [36][37][38][39]. Each of them tries to show the channel characteristics that are more representative of the objectives pursued by each researcher. The underwater acoustic channels require particular attention on several particular aspects that belong to aquatic environments. In the section ?? it will be presented some relevant channel models.

5.2 Linear Systems

A *linear system* is a mathematical abstraction of a system using linear operators. Linear systems regularly exhibit behavior and properties that are simpler than non-linear systems. As a mathematical model or idealization, linear systems find important applications in control theory, signal processing, and communications. The propagation medium for wireless communication systems can be modeled by linear systems or combinations of them.

Let \mathcal{T} be a linear system such $y_1(t) = \mathcal{T}\{x_1(t)\}$, and $y_2(t) = \mathcal{T}\{x_2(t)\}$, then must satisfy

$$\alpha y_1(t) + \beta y_2(t) = \mathcal{T}\{\alpha x_1(t) + \beta x_2(t)\}, \quad (5.1)$$

for any scalar constants α , and β .

The operation of the resulting system depends of the input and it can be described as a sum of responses to simpler inputs. For time-invariant systems, it is the basis of the impulse response or the frequency response methods, which describe a general input function $x(t)$ in terms of unit impulses or frequency components. A linear system can be characterized through of its impulse response ($h(t)$) or its frequency response ($H(f)$).

Most practical systems are studied using linear models. If the nature of the system is intrinsically non-linear then it is possible fragment it for analyzing linear segments in each stage.

5.3 Stochastic Systems

The word *stochastic* became of a Greek root that means *pertaining to chance*. This term is used to describe systems that present random or stochastic behavior. This work deals with systems which one or more parts of them have randomness associated with them. Unlike a deterministic system, a stochastic system does not always produces the same output for a given input. A few components of systems that can be stochastic in nature include stochastic inputs, random time-delays, noisy disturbances, and even stochastic dynamic processes.

Stochastic systems is a branch of systems theory that deals with dynamic and static systems, whose behavior are characterized by probability distributions or spectral measures. Stochastic systems is applied in various areas within science, such as control, communications and networks, signal processing, biology or finance. Many of the mathematical abstractions employed within the theoretical framework of stochastic systems were originated in an attempt to measure in an accurate manner the treatment of probabilistic modeling and inference, random dynamical systems, and information. This discipline was established on the axiomatic

approach to probability of *Kolmogorov*, the random noise model of *Wiener* and the information measure of *Shannon*.

5.4 Time-Variant Systems

A *Linear Time-Variant* (LTV) system is a linear system whose behavior changes over time. These systems are not *time-invariant* systems but if their rate of temporal change is negligible during some period of time, then we can consider them time-invariant systems for analytic purposes. In this category certain parameters governing the system's behavior change with time, so that the system could respond differently to the same input at different times.

There are many well understood procedures for dealing with the response of linear time invariant systems, such as Fourier transforms. However, these procedures are not valid for time-variant systems. When a system undergoing very slow time variation in comparison to its time constants can usually be considered to be time invariant for analytic purposes. These systems are close to time invariant on a small scale (*coherent time*). An example of a time-variant system, that can be tried as time-invariant, is a double dispersive channel, which has a time-invariant behavior during a coherent time t_C . Thus does not result in any behavior qualitatively different from that observed in a time invariant system: t_C -to- t_C , they are effectively time invariant, though t_C to t_C , the parameters may change. Other linear time variant systems may behave more like non-linear systems, if the system changes quickly and significantly differently between measurements. The following characteristics can be associated to time-variant systems:

- They are time-dependent.
- They do not have an impulse response $h(t)$ in the normal sense. The system can be characterized by an impulse response except the impulse response must be known at each and every time interval.

- They are not stationary.

5.5 Study of Communication Systems

Both, electromagnetic and acoustic channels can be modeled as linear time-variant systems, therefore we will start with the classical models developed for linear time-variant (LTV) systems. On this point is mandatory to reference Bello works. Bello offered big contributions in the field of linear time-variant systems characterization. So, its work is fundamental in channel characterization research area [15].

Table 5–1: Bello’s Kernel Functions

Input/Output	Time	Frequency
Time	$K_1(t, s)$	$K_4(f, t)$
Frequency	$K_3(t, f)$	$K_2(f, l)$

Bello presents a dual classification of the time-frequency functions for time-variant systems characterization. Using the filter perspective, Bello formulates the time-variant input-output relation at time-variant system in terms of four distinct kernels, acting on the input signal $z(t)$ in time domain or $Z(f)$ in frequency domain, both representations are equivalent (dual representations). Generated output is the signal $w(t)$ in time domain or $W(f)$ in frequency domain. The four kernels are presented in the Table 5–1.

The input-output relationships are presented now,

$$w(t) = \int z(s)K_1(t, s)ds \quad (5.2)$$

$$W(f) = \int Z(l)K_2(f, l)dl \quad (5.3)$$

$$w(t) = \int Z(f)K_3(t, f)df \quad (5.4)$$

$$W(f) = \int z(t)K_4(f, t)dt \quad (5.5)$$

Starting in the relationship (5.2) and doing the variable change $s = t - \xi$ we obtain

$$\begin{aligned} w(t) &= \int z(t - \xi)K_1(t, t - \xi)d\xi \\ w(t) &= \int z(t - \xi)g(t, \xi)d\xi, \end{aligned} \quad (5.6)$$

where $g(t, \xi) = K_1(t, t - \xi)$ is known as *input-delay spread function* because the first operation is input-delay and the second operation is the complex exponential product. On the other hand, if the first operation is the complex exponential product and the second operation is the output-delay then, we obtain other input-output relation which it is presented as follows:

$$w(t) = \int z(t - \xi)h(t - \xi, \xi)d\xi, \quad (5.7)$$

where $h(t, \xi) = K_1(t + \xi, t)$ is known as the *output-delay spread function*. Equation (5.6) is fundamental in this work. The kernel $K_1(t, s)$ is known as *time-variant impulse response*.

Applying the time-frequency duality principle developed by Bello we can obtain a complete set of time-frequency characterization functions for linear time-variant channels [40]. All these functions allow us to visualize the channel in some particular perspective, aiding in the modeling and estimation processes. The following are the more important time-frequency system functions for linear time-variant channels.

Let $H(f, \nu) = K_2(f, f - \nu)$ and $G(f, \nu) = K_2(f + \nu, f)$ be variable substitutions, then

$$W(\nu) = \int Z(f - \nu)H(f, \nu)df \quad (\text{freq-freq}) \quad (5.8)$$

$$W(f) = \int Z(f - \nu)G(f, \nu)d\nu \quad (\text{freq-freq}), \quad (5.9)$$

where $H(f, \nu)$ and $G(f, \nu)$ are known as *input-Doppler spread function* and *output-Doppler spread function* respectively. These functions are *dual functions* of *input-delay spread function* $g(t, \xi)$ and *output-delay spread function* $h(t, \xi)$. The kernel $K_2(f, l)$ is known as *bi-frequency function*.

In the same manner, the kernels $K_3(t, f)$ and $K_4(f, t)$ can be transformed to $K_3(t, f) = e^{-j2\pi ft}T(f, t)$ and $K_4(f, t) = e^{+j2\pi ft}M(t, f)$, and then, replaced in equations (5.4) and (5.5), where it is possible appreciate the same time-frequency dual treatment applied to $K_1(t, s)$ and $K_2(f, l)$, we can obtain the following input-output relations

$$w(t) = \int Z(f)T(f, t)e^{+j2\pi ft}df \quad (\text{freq-time}) \quad (5.10)$$

$$W(f) = \int z(t)M(t, f)e^{-j2\pi ft}dt \quad (\text{time-freq}), \quad (5.11)$$

where $T(f, t)$ and $M(t, f)$ are known as *time-variant transfer function* and *frequency-dependent modulation function*, and these are time-frequency dual functions.

Using as reference Equations (5.9) and (5.10) and applying the time-frequency duality principle, it is possible to establish a relation between the *time-variant transfer function* $T(f, t)$ and *output-Doppler spread function* $G(f, \nu)$ as follow

$$T(f, t) = \int G(f, \nu)e^{+j2\pi \nu t}d\nu. \quad (5.12)$$

Similarly combining Equations (5.7) and (5.11) it is possible to establish a relation between the *frequency-dependent modulation function* $M(t, f)$ and the *output-delay spread function* $h(t, \xi)$ as follows:

$$M(t, f) = \int h(t, \xi) e^{-j2\pi f t} d\xi. \quad (5.13)$$

Finally, two very important time-frequency characterization functions for linear time-variant channels are presented. These functions are the *delay-Doppler spread function* $U(\xi, \nu)$ and the *Doppler-delay spread function* $V(\nu, \xi)$. Both establishes very important input-output relations for linear time-variant channels. These functions are defined as:

$$U(\xi, \nu) = \int g(t, \xi) e^{-j2\pi \nu t} dt \quad (5.14)$$

$$V(\nu, \xi) = \int H(f, \nu) e^{+j2\pi \xi f} df \quad (5.15)$$

or in equivalent manner

$$g(t, \xi) = \int U(\xi, \nu) e^{+j2\pi \nu t} d\nu \quad (5.16)$$

$$H(f, \nu) = \int V(\nu, \xi) e^{-j2\pi \xi f} d\xi, \quad (5.17)$$

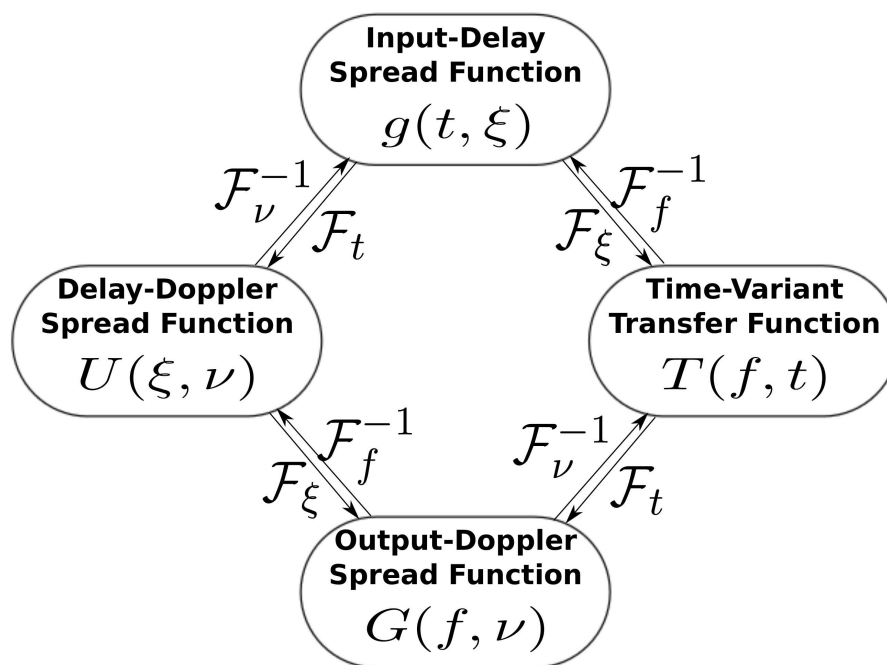
where $g(t, \xi)$ and $H(f, \nu)$ are the *input-delay spread function* and the *input-Doppler spread function* respectively. Substituting Equations (5.16) and (5.17) in Equations (5.6) and (5.8) respectively is possible to obtain the following input-output relations

$$w(t) = \int \int z(t - \xi) U(\xi, \nu) e^{+j2\pi \nu t} d\nu d\xi \quad (5.18)$$

$$W(f) = \int \int Z(f - \nu) V(\nu, \xi) e^{-j2\pi \xi f} d\xi d\nu. \quad (5.19)$$

Figure (5-1) shows the relationship between the main time-frequency characterization functions for linear time-variant (LTV) channels. The direct Fourier operator is denoted using the symbol \mathcal{F}_v with a subscript v that denote the variable on the

transformation. Similarly the symbol \mathcal{F}_v^{-1} denote the inverse Fourier operator and the subscript v denote the variable on the transformation.



Duality Property

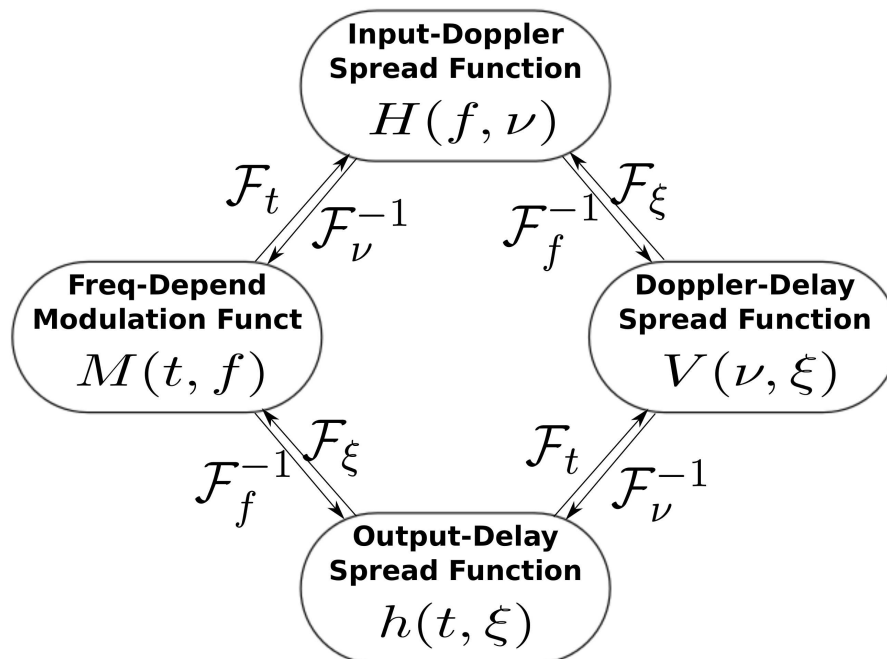


Figure 5–1: Time-Frequency System Functions for LTV Channels

Other very important relationship must be clearly established between the *input-delay spread function* $g(t, \xi)$, the *delay-Doppler spread function* $U(\xi, \nu)$, and their duals functions the *output-Doppler spread function* $G(f, \nu)$, and the *Doppler-delay spread function* $V(\nu, \xi)$. Figure 5–2 illustrates this relation.

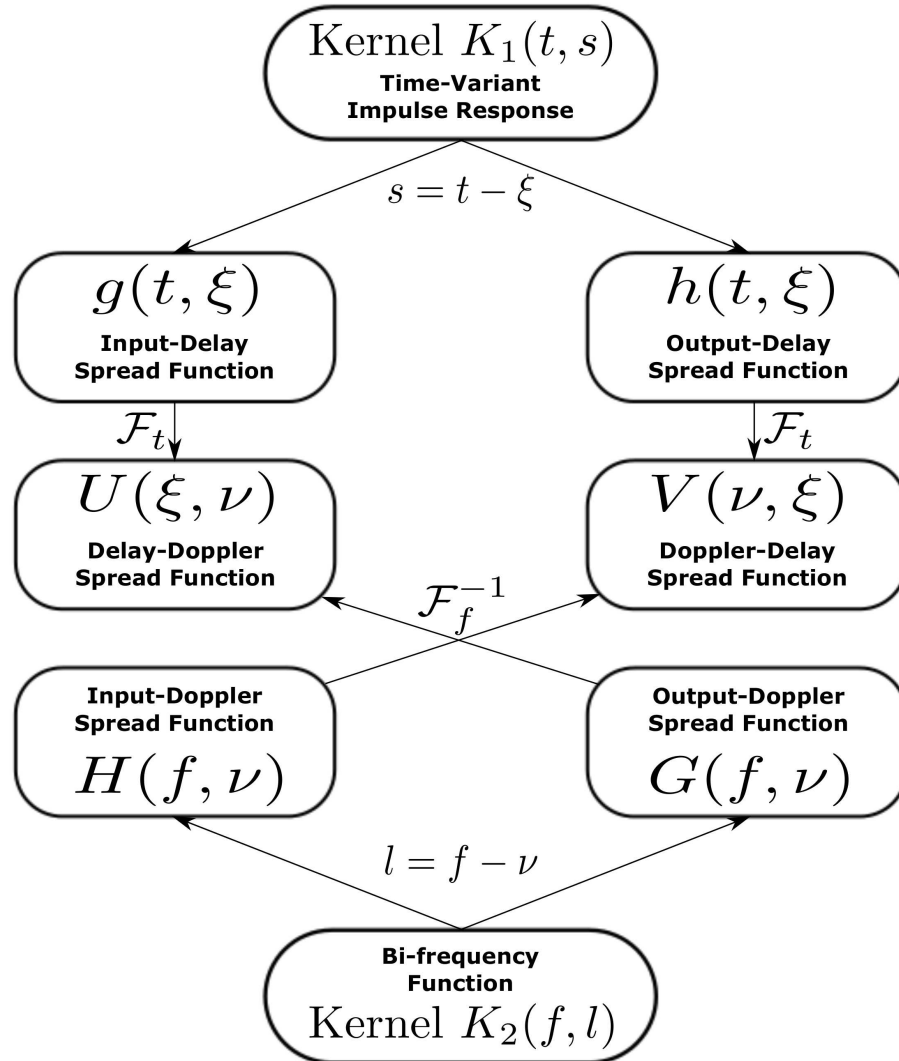


Figure 5–2: Relationship Between Delay-Doppler Spread Function and the Kernels $K_1(t, s)$ and $K_2(f, l)$

The relationship between system functions was extended to express the Fourier relationship existing between three of most important bi-dimensional functions. This time-frequency functions are widely discussed in the scientific literature. They are the *Ambiguity Function* (A_x), the *Wigner Distribution* (W_x), and the *Correlation*

Function (R_x). Figure 5–3 illustrates the relationship between the A_x , R_x , and W_x functions. This relationship was used as starting point, in this thesis work, to establish a new extended relation that include the *Delay-Doppler Spread Function* (U_X). The purpose to extend this relation is to create the basis to use the *Delay-Doppler Spread Function* (U_x , known in some articles as *Scattering Function*) as surrogate estimation function instead as the impulse response $h(t)$ (or its equivalent LTV *Input-Delay Spread function* ($g(t, \tau)$). The reason behind this change is sustained by the fact of the functions $h(t)$ or $g(t, \tau)$ are time-dependent only. This function don't consider the Doppler effect on the channel behavior. So, we need to introduce a new “surrogate” function that considers the Doppler effect in its mathematical description. This function is the “scattering function”. It is represented in this thesis work by the *Delay-Doppler Spread Function* $U_x(\nu, \xi)$.

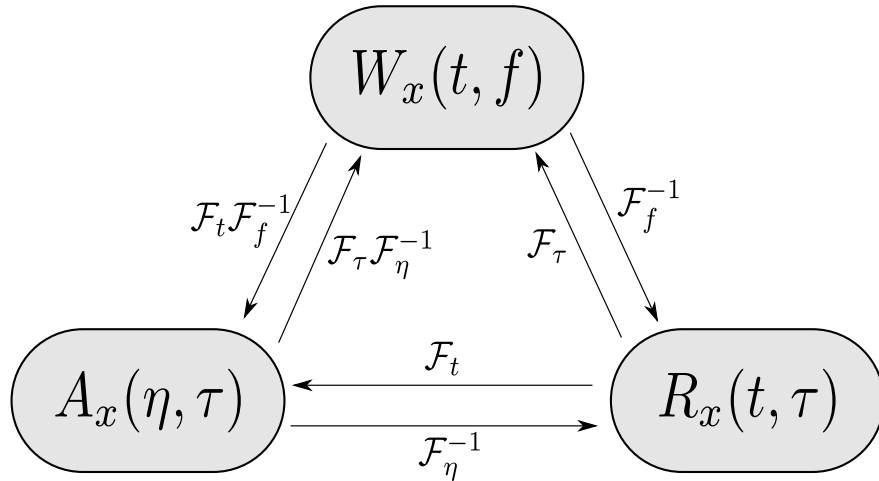


Figure 5–3: **Relationship Between the Ambiguity Function, the Wigner Distribution, and the Correlation Function**

Inside the estimation process the main function used to get the channel parameters was the *Delay-Doppler Spread Function* $U_x(\nu, \xi)$. This function establishes a binding relationship between the time and frequency behaviors in a *MIMO ALS Channel*. Figure 5–4 illustrates the new extended relationship, presented in this

work as a original contribution, between the *Ambiguity Function*, the *Wigner Distribution*, the *Correlation Function*, and the *Delay-Doppler Spread Function*.

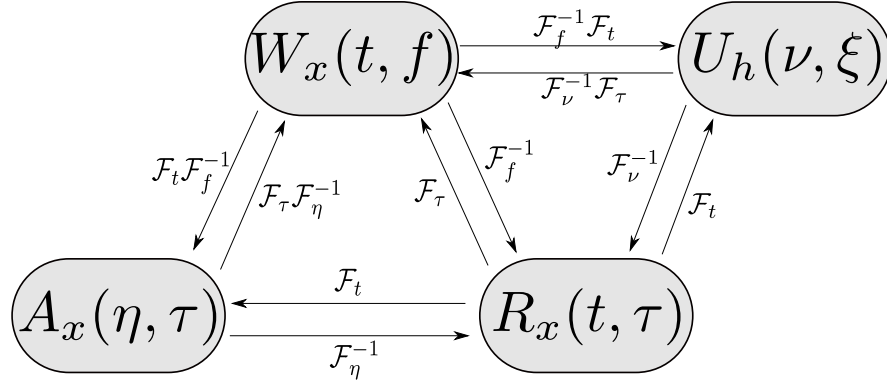


Figure 5–4: **Enhanced Diagram Depicting Relationship Between the Ambiguity Function, the Wigner Distribution, the Correlation Function, and the Delay-Doppler Spread Function $U(\nu, \xi)$.**

5.6 Estimation Using a Parallel Approach

5.6.1 The Channel as an Operator

An acoustic channel can be modeled as an acoustic linear stochastic (ALS) system. Under a discrete perspective both, the input signal $z[m]$ and the output signal $w[m]$ belong to a signals space denoted as $l^2(\mathcal{Z}_M)$, where M is the length of the signals and these can be represented using one-dimensional vectors.

In terms of vector spaces, an operator is a mapping from one vector space or module to another. Considering an ALS channel as a mapping from an input signal z to an output signal w is clear to intuit that the action of the channel on the input signal for mapping to the output signal can be modeled as an operator. Having defined the ALS channel as a linear system we can infer that the channel can be represented as a linear operator. This quality gives the channel a number of features and very desirable properties for easier analysis.

We denote now the ALS channel as the continuous linear system \mathcal{T} , and the channel input-output relationship will be expressed as:

$$w(t) = \mathcal{T}\{z(t)\}, \quad t \in \mathbb{R}, w, z \in l^2(\mathbb{R}), \quad (5.20)$$

where \mathcal{T} is a linear system, representing the action of an ALS channel on the input signal $z(t)$ for producing the output signal $w(t)$, and $l^2(\mathcal{R})$ is a signals space of all continuous signals with finite energy.

The more important consideration when we accept the channel as a discrete linear operator is the possibility of representation using a finite dimensional matrix \mathbf{H} , allowing to develop computational approaches for channel modeling and estimation, exploiting the fast algorithms for matrix treatment. This consideration is essential in this work, and the SISO (single-input single-out) ALS channel behavior under operators theory perspective is modeled as:

$$\mathbf{w} = \mathbf{H}\mathbf{z} + \mathbf{n}, \quad (5.21)$$

where \mathbf{n} , \mathbf{w} , and \mathbf{z} are one-dimensional vectors belong to the signals space $l^2(\mathcal{Z}_M)$ and represent *noise*, *output signal*, and *input signal* respectively; and \mathbf{H} represents the channel matrix.

5.6.2 Characterization Function for an ALS channel

In the case of linear time-invariant (LTI) systems, the *impulse response* $h(t)$ characterize them in unequivocal manner. By definition $h(t)$ is the system response to the $\delta(t)$, known as *impulse function*. So,

$$h(t) = \mathcal{T}\{\delta(t)\}, \quad t \in \mathbb{R}, h \in l^2(\mathbb{R}). \quad (5.22)$$

The output of a LTI system is given by:

$$w(t) = \int_{-\infty}^{\infty} z(\xi)h(t - \xi)d\xi, \quad t, \xi \in \mathbb{R}; w, z \in l^2(\mathbb{R}), \quad (5.23)$$

where the integral operation is known as convolution and the integration variable ξ represents a temporal displacement, shift or delay.

The convolution operation admits commutative property. Another manner to write the Equation (5.23) is

$$w(t) = (z * h)(t), \quad t \in \mathbb{R}, w, z \in l^2(\mathbb{R}). \quad (5.24)$$

The ALS channel input-output relationship is given by:

$$w(t) = \int_{-\infty}^{\infty} z(t - \xi)g(t, \xi)d\xi, \quad t, \xi \in \mathbb{R}, z, g, w \in l^2(\mathbb{R}). \quad (5.25)$$

In this case, the *input-delay spread function* $g(t, \xi)$ is a function of time t and delay ξ , for adapting to time-variant system behavior, so the new characterization function $g(t, \xi)$, for ALS systems is formulated as:

$$h(t, \xi) = \mathcal{T}\{\delta(t - \xi)\}, \quad t, \xi \in \mathbb{R}, h \in l^2(\mathbb{R}). \quad (5.26)$$

Using the Bello nomenclature we call the *kernel function* $h(t, \xi)$ the *time-variant impulse response* [15]. But we will substitute the *kernel function* $h(t, \xi)$ by $g(t, \xi)$ and we will call it *input-delay spread function*, that will be the ALS channel characterization function.

Figure (5-5) shows the characterization functions $h(t)$ and $g(t, \xi)$ for LTI and ALS systems.

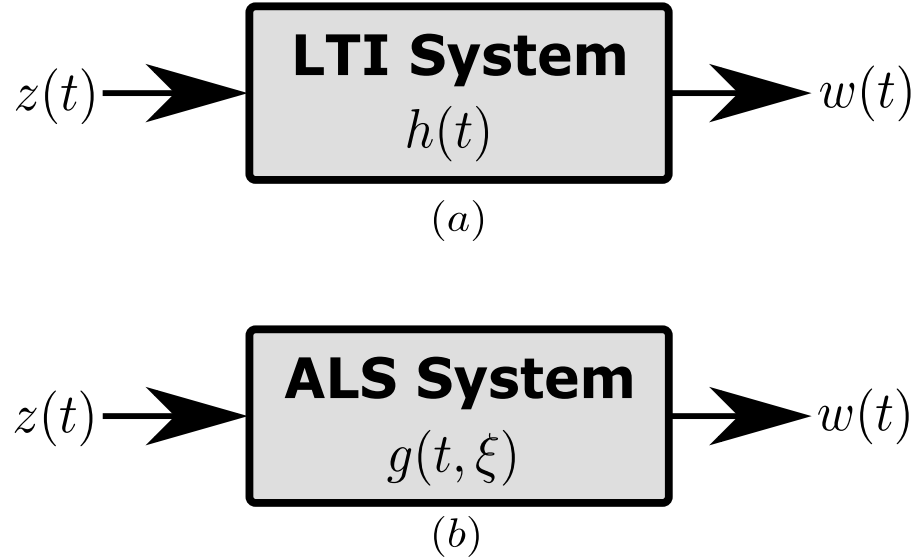


Figure 5–5: Functions $h(t)$ vs. $g(t, \xi)$: (a) LTI Systems (b) ALS Systems

In the spectral domain the Equation (5.24) can be reformulated as:

$$W(\omega) = \mathcal{F}\{(z * h)(t)\} = Z(\omega)H(\omega),$$

$$\omega \in \mathbb{R}, W, H, Z \in l^2(\mathbb{R}), \quad (5.27)$$

where \mathcal{F} represents the Fourier operator and $W(\omega)$, $Z(\omega)$ and $H(\omega)$ are the Fourier transforms of $w(t)$, $z(t)$ and $h(t)$ respectively. Equations (5.24) and (5.27) represent the convolution theorem, that shows the duality of convolution and Hadamard product in time and frequency domains.

5.6.3 Coherence Time T_C and Coherence Bandwidth B_C in the Channel Modeling

In the practice, although one ALS channel to be considered a time variant system, in where the characterization function $g(t, \xi)$ change over the time, the more adequate approach for modeling and estimation considerations is assume that the ALS channel is *time-invariant* in a time period called *coherence time* (T_C). During the *coherence time* the parameter values of the *input-delay spread function* $g(t, \xi)$ stay constants. This assumption allows to resolve the input-output relationship as it is expressed in the (5.25) using a convolution operation.

The ALS channels are considered doubly dispersive because their behavior change in time and in frequency. In dual manner the ALS channel can be treated as LTI channel if the frequency changes do not exceed a specific value called *coherence bandwidth* B_C .

The assumption of the existence of a *coherence time* T_C and a *coherence bandwidth* B_C leads to the problem of establishing as accurate as possible, what is the value of those *coherence time* and *coherence bandwidth*. These parameters regulate the maximum amount of samples that can be processed using a constant *input-delay spread function* $g(t, \xi)$. In ALS channels the principal considerations for calculate these parameters are associated with the propagation speed of the sound in the water (approx. $1,500 \frac{mts}{seg}$), the effect of the high frequencies in the underwater propagation of the acoustic waves and other factors as signal to noise ratio (SNR).

Beaujean and LeBlanc analyzed the relation between the *coherence time* and *coherence bandwidth* in the context of channel characterization and underwater acoustic systems classification [41]. Two main types of fading channels can be classified using the TF product (time spread and frequency spread) of the channel. When this product is greater than 1 then the channel is considered an *overspread channel* and otherwise the channel is considered an *under-spread channel*. This classification allows to do previsions in order to adequately model to the nature of the different channels.

5.7 Implementation Results

The following graphs describe implementation results obtained through the CSP modeling of imaging sonar and scattering sub-systems when interacting with diverse pulse waveforms, in particular, rectangular pulses, sinusoidal pulses, and linear chirp pulses.

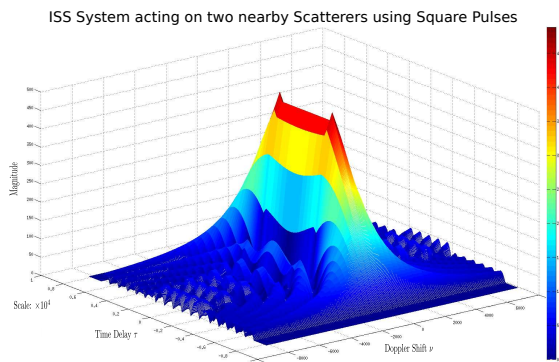


Figure 5–6: ISS System: Two Nearby Scatterers using Square Pulses.

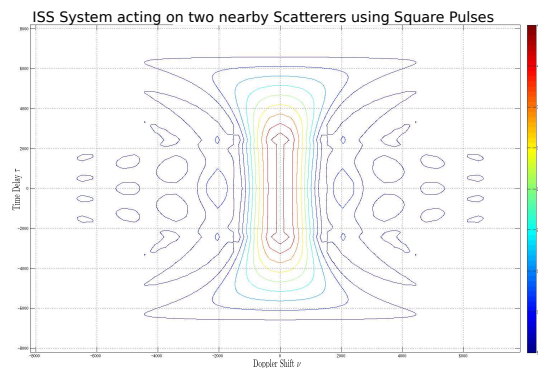


Figure 5–7: ISS System: Two Nearby Scatterers using Square Pulses.

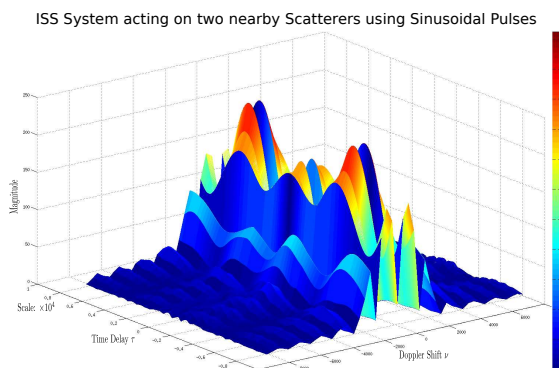


Figure 5–8: ISS System: Two Nearby Scatterers using Sinusoidal Pulses.

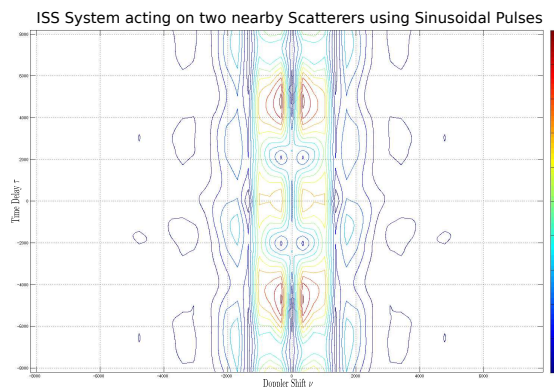


Figure 5–9: ISS System: Two Nearby Scatterers using Sinusoidal Pulses.

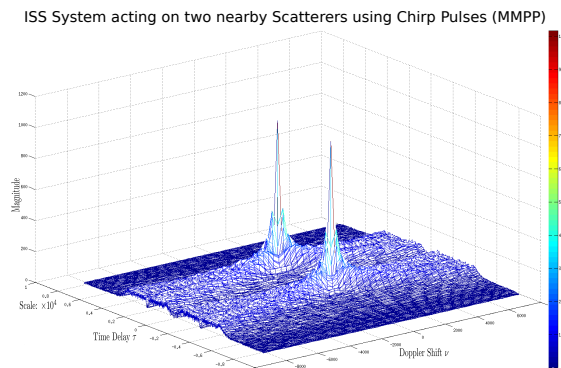


Figure 5–10: ISS System: Two Nearby Scatterers using Optimized Chirp Pulses (MMPP).

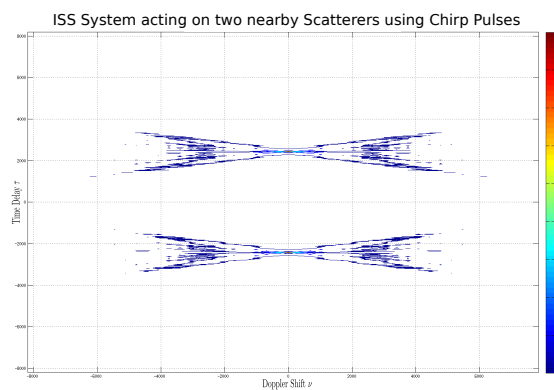


Figure 5–11: ISS System: Two Nearby Scatterers using Optimized Chirp Pulses (MMPP).

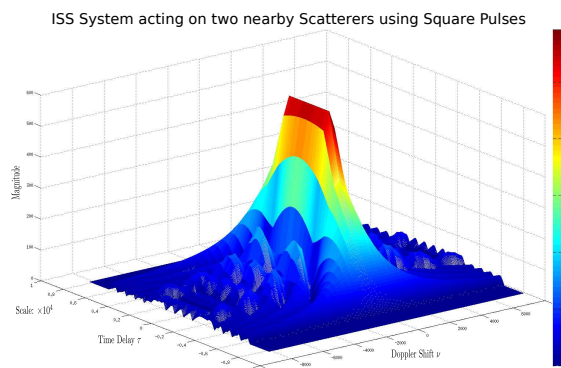


Figure 5–12: ISS System: Two Nearby Scatterers using Square Pulses.

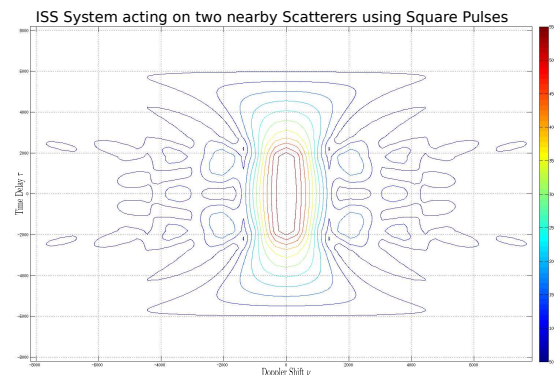


Figure 5–13: ISS System: Two Nearby Scatterers using Square Pulses.

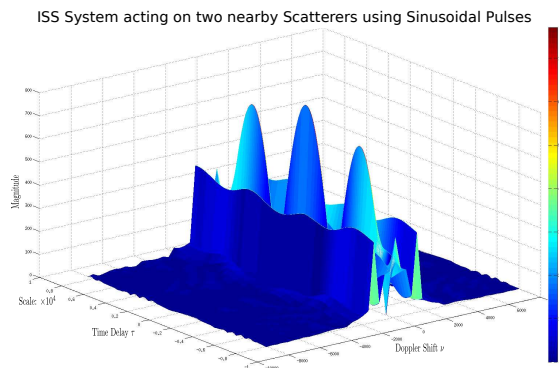


Figure 5–14: ISS System: Two Nearby Scatterers using Sinusoidal Pulses.

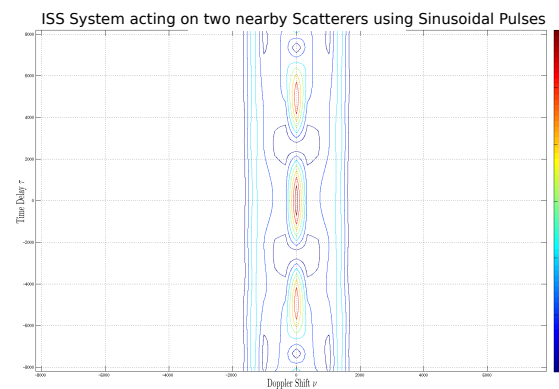


Figure 5–15: ISS System: Two Nearby Scatterers using Sinusoidal Pulses.

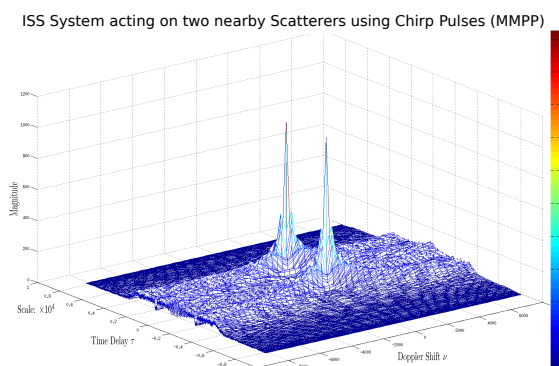


Figure 5–16: ISS System: Two Nearby Scatterers using Optimized Chirp Pulses (MMPP).

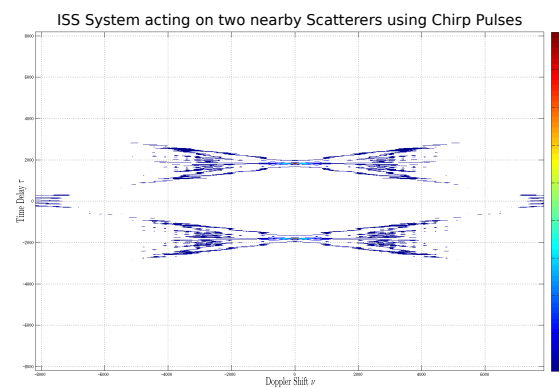


Figure 5–17: ISS System: Two Nearby Scatterers using Optimized Chirp Pulses (MMPP).

6. MIMO Channel Parameter Estimation Algorithms Implementation

6.1 Introduction

In this chapter we present the work associated with the *matching pursuit* Greedy algorithm implementation to estimate MIMO Channel parameters. The characterization channel function used in this thesis is the *Delay-Doppler Spread Function* $\mathbf{U}(\tau, \nu)$. This function plays the role of surrogate function of the *Input-Delay Spread Function* that is the *impulse response* of the acoustic linear stochastic (ALS) channels. The parameters to be estimated are the time-delays and the Doppler-shift associated with the Delay-Doppler Spread Function. This estimation work must be carried out in a fraction of time correspondent to the *coherence time* T_C , time in which the *impulse response* functions is assumed as constant. Therefore, the estimation of these parameters constitutes a fundamental stage in communication and sonar problems.

6.2 Channel Configurations

To analyze this problem we start by presenting the **SISO**, **MISO**, and **SIMO** problems. The idea behind this approach is to visualize the algebraic structures in each case, for generalizing the structures in the **MIMO** case. Each problem has a particular matrix-vector formulation. These formulations allow us to develop data structures and procedures appropriate to address the efficient estimation of the problem. Another important objective in this chapter is associated with establishing the tensor matrix structures. These structures open an interesting field of study. Regularities in the matrix structures can be exploited for improving the algorithms. This thesis exploits the matrix-vector formulations and sparsity properties, but many other regularities can be reviewed and known to develop new algorithms.

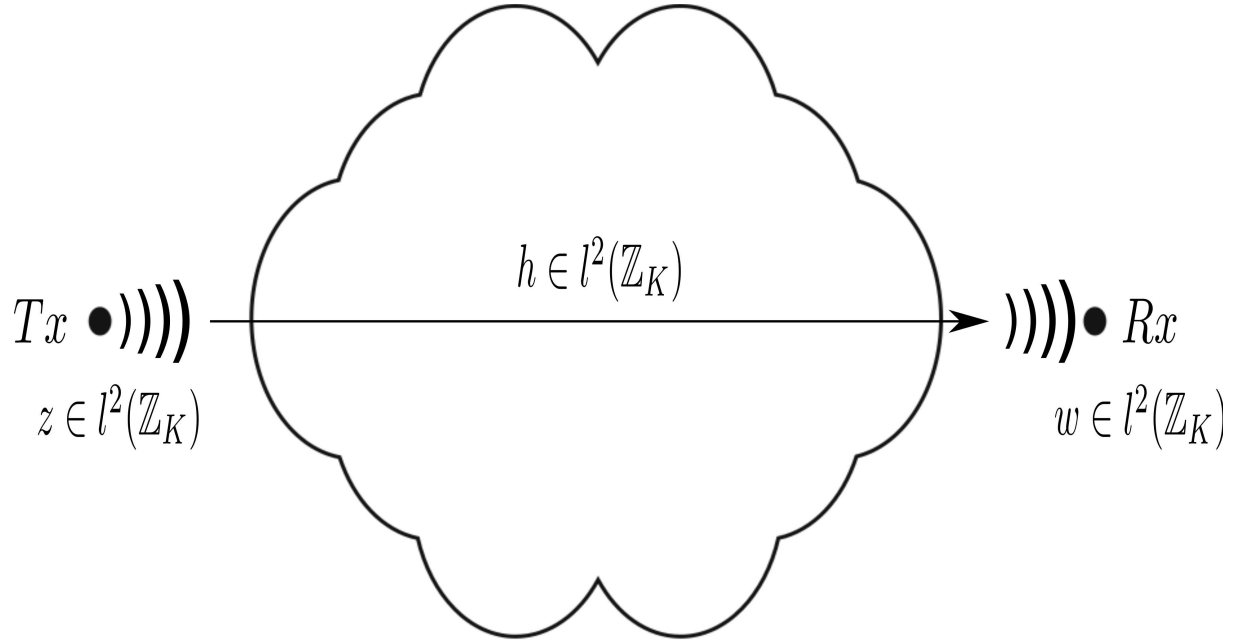


Figure 6–1: Schematic of SISO Channel System

6.2.1 SISO Case

SISO case has just one transmitter transducer and one receiver transducer. Equation (6.1) shows the input-output relation in the SISO case.

$$w = z \circledast_K h = \mathcal{C}\{z\}h = \mathbf{Z}h, \mathbf{Z} \in l^2(\mathbb{Z}_K \times \mathbb{Z}_K), \quad (6.1)$$

where $z \in l^2(\mathbb{Z}_K)$ is the transmitted signal, $h \in l^2(\mathbb{Z}_K)$ is the *impulse response* of the channel, $w \in l^2(\mathbb{Z}_K)$ is the received signal, \circledast_K is the cyclic convolution operator of order K , and \mathcal{C} is the circulant operator. Equation 6.1 shows a typical input-output relation in a linear and time-invariant system (a filter).

Figure 6–1 illustrates a SISO system. The matrix-vector representation of the problem is presented in Equation (6.2) which shows the first relevant matrix structure known as *circulant matrix*

$$\begin{bmatrix} w[0] \\ w[1] \\ \vdots \\ w[L-2] \\ w[L-1] \end{bmatrix} = \begin{bmatrix} z[0] & z[L-1] & \dots & z[1] \\ z[1] & z[0] & \dots & z[2] \\ \vdots & \vdots & \ddots & \vdots \\ z[L-2] & z[L-3] & \dots & z[L-1] \\ z[L-1] & z[L-2] & \dots & z[0] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[L-2] \\ h[L-1] \end{bmatrix}. \quad (6.2)$$

In training mode, some series of predefined pulses z are transmitted. These pulses are compared with the received signal w for characterizing the **SISO** channel using the inverse relation shown in Equation (6.3).

$$h = \mathbf{Z}^\dagger w, \quad (6.3)$$

where the vector $h \in l^2(\mathbb{Z}_K)$, the matrix $\mathbf{Z} \in l^2(\mathbb{Z}_K \times \mathbb{Z}_K)$, and the vector $w \in l^2(\mathbb{Z}_K)$.

The computational complexity, under direct computation approach, associated with this problem is $\mathbf{O}(K^3)$ and it is given by the complexity of the inverse operator acting over the matrix \mathbf{Z} of dimension $K \times K$, and the complexity of the matrix-vector product of the vector w of length K . Therefore, the dominant complexity is $\mathbf{O}(K^3)$, where K is the number of samples in the impulse response $h(t)$.

6.2.2 MISO Case

MISO case has M transmitter transducers and just one receiver transducer. Equation (6.4) shows the input-output relation in the MISO case.

$$w = \sum_{i \in \mathbb{Z}_M} z_i \otimes_K h_i = \left(\bigsqcup_{i \in \mathbb{Z}_M} \mathcal{C}\{z_i\} \right) \left(\bigvee_{i \in \mathbb{Z}_M} h_i \right) = \mathbf{Z} \mathbf{h}, \quad (6.4)$$

where $z_i \in l^2(\mathbb{Z}_K)$ is the i -th transmitted signal, $h_i \in l^2(\mathbb{Z}_K)$ is the i -th *impulse response*, $w \in l^2(\mathbb{Z}_K)$ is the received signal, \otimes_K is the cyclic convolution operator

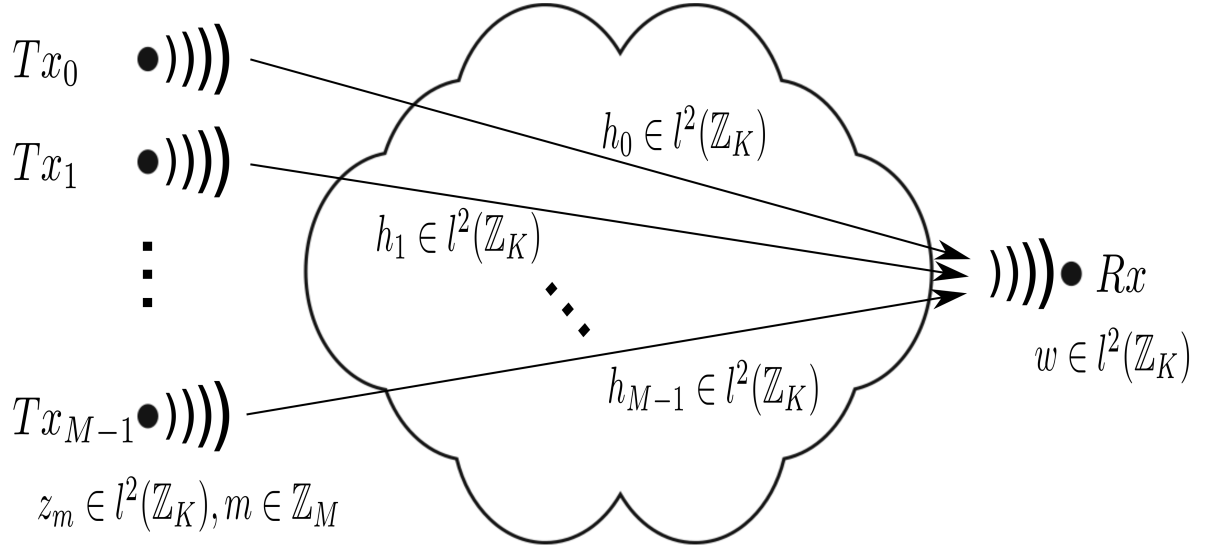


Figure 6–2: Schematic of MISO Channel System

of order K , \sqcup is the horizontal matrix concatenation operator, \vee is the vertical matrix concatenation operator, $\mathbf{Z} \in l^2(\mathbb{Z}_K \times \mathbb{Z}_{KM})$ is the input channel matrix, $\mathbf{h} \in l^2(\mathbb{Z}_{KM})$ is the impulse response channel vector, and \mathcal{C} is the circulant operator.

Figure 6–2 illustrates the MISO system. A matrix-vector representation is shown in Equation (6.5).

$$\begin{bmatrix} w[0] \\ w[1] \\ \vdots \\ w[L-2] \\ w[L-1] \end{bmatrix} = \begin{bmatrix} z_0[0] & \dots & z_0[1] & \dots & z_{M-1}[0] & \dots & z_{M-1}[1] \\ z_0[1] & \dots & z_0[2] & \dots & z_{M-1}[1] & \dots & z_{M-1}[2] \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ z_0[L-2] & \dots & z_0[L-1] & \dots & z_{M-1}[L-2] & \dots & z_{M-1}[L-1] \\ z_0[L-1] & \dots & z_0[0] & \dots & z_{M-1}[L-1] & \dots & z_{M-1}[0] \end{bmatrix} \begin{bmatrix} h_0[0] \\ \vdots \\ h_0[L-1] \\ \vdots \\ h_{M-1}[0] \\ \vdots \\ h_{M-1}[L-1] \end{bmatrix}. \quad (6.5)$$

In training mode, some series of predefined pulses z are transmitted. These pulses are compared with the received signal w for characterizing the **MISO** channel

using the inverse relation shown in Equation (6.6).

$$\mathbf{h} = \mathbf{Z}^\dagger w, \quad (6.6)$$

where $\mathbf{Z} \in l^2(\mathbb{Z}_K \times \mathbb{Z}_{KM})$ is the input channel matrix, and $\mathbf{h} \in l^2(\mathbb{Z}_{KM})$ is the impulse response channel vector.

The computational complexity, under direct computation approach, associated with the MISO estimation problem is $\mathbf{O}((KM)^3 + 2K^3M^2) = \mathbf{O}((KM)^3)$, and it is given by the complexity of the pseudo-inverse operator acting over the matrix \mathbf{Z} of dimension $K \times KM$, and the complexity of the matrix-vector product of the vector \mathbf{h} of length KM . Therefore, the dominant complexity is $\mathbf{O}((KM)^3)$, where K is the number of samples in the impulse response $h(t)$, and M is the number of transmitter transducers.

6.2.3 SIMO Case

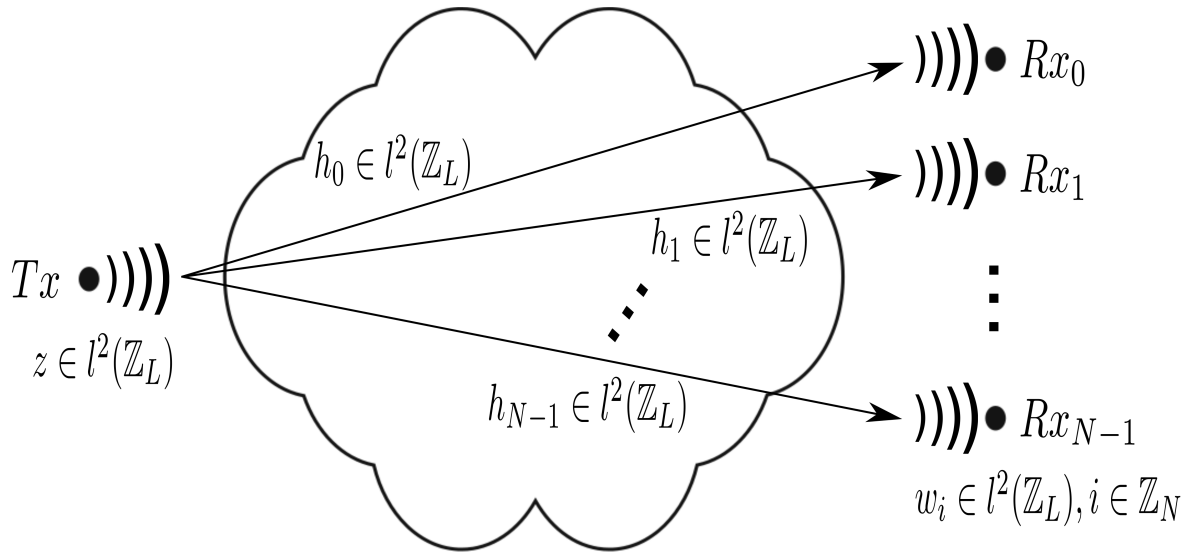


Figure 6–3: Schematic of SIMO Channel System

SIMO case has just one transmitter transducer, but N receiver transducers. Equation (6.7) shows the input-output relation in the SIMO channel system.

$$\mathbf{w} = \bigvee_{i \in \mathbb{Z}_N} z \circledast_K h_i = \left(\bigoplus_{i \in \mathbb{Z}_N} \mathcal{C}\{z\} \right) \left(\bigvee_{i \in \mathbb{Z}_N} h_i \right) = (I_N \otimes \mathcal{C}\{z\}) \left(\bigvee_{i \in \mathbb{Z}_N} h_i \right) = \mathbf{Z}\mathbf{h}, \quad (6.7)$$

where $z \in l^2(\mathbb{Z}_K)$ is the transmitted signal, $h_i \in l^2(\mathbb{Z}_K), i \in \mathbb{Z}_N$ is the i -th *impulse response*, $w_i \in l^2(\mathbb{Z}_K), i \in \mathbb{Z}_N$ is the i -th received signal, \circledast_K is the cyclic convolution operator of order K , \bigvee is the vertical matrix concatenation operator, $\mathbf{Z} \in l^2(\mathbb{Z}_{KN} \times \mathbb{Z}_{KN})$ is the input channel matrix, $\mathbf{h} \in l^2(\mathbb{Z}_{KN})$ is the impulse response channel vector, $\mathbf{w} \in l^2(\mathbb{Z}_{KN})$ is the output vector, and \mathcal{C} is the circulant operator.

Figure 6-3 illustrates the SIMO system. A matrix-vector representation is shown in Equation (6.8).

$$\begin{bmatrix} w_0[0] \\ w_0[1] \\ \vdots \\ w_0[L-2] \\ w_0[L-1] \\ \vdots \\ w_{N-1}[0] \\ w_{N-1}[1] \\ \vdots \\ w_{N-1}[L-2] \\ w_{N-1}[L-1] \end{bmatrix} = \begin{bmatrix} z[0] & z[L-1] & \dots & z[1] & \dots & 0 & 0 & \dots & 0 \\ z[1] & z[0] & \dots & z[2] & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \dots & \vdots & \vdots & \ddots & \vdots \\ z[L-2] & z[L-3] & \dots & z[L-1] & \dots & 0 & 0 & \dots & 0 \\ z[L-1] & z[L-2] & \dots & z[0] & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \dots & z[0] & z[L-1] & \dots & z[1] \\ 0 & 0 & \dots & 0 & \dots & z[1] & z[0] & \dots & z[2] \\ \vdots & \vdots & \ddots & \vdots & \dots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & z[L-2] & z[L-3] & \dots & z[L-1] \\ 0 & 0 & \dots & 0 & \dots & z[L-1] & z[L-2] & \dots & z[0] \end{bmatrix} \begin{bmatrix} h_0[0] \\ h_0[1] \\ \vdots \\ h_0[L-2] \\ h_0[L-1] \\ \vdots \\ h_{N-1}[0] \\ h_{N-1}[1] \\ \vdots \\ h_{N-1}[L-2] \\ h_{N-1}[L-1] \end{bmatrix}. \quad (6.8)$$

In training mode, some series of predefined pulses z are transmitted. These pulses are compared with the received signals $w_i, i \in \mathbb{Z}_N$ for characterizing the

SIMO channel using the inverse relation showed in Equation (6.9).

$$\mathbf{h} = \mathbf{Z}^\dagger \mathbf{w}, \quad (6.9)$$

where $\mathbf{Z} \in l^2(\mathbb{Z}_{KN} \times \mathbb{Z}_{KN})$ is the input channel matrix, $\mathbf{h} \in l^2(\mathbb{Z}_{KN})$ is the impulse response channel vector, and $\mathbf{w} \in l^2(\mathbb{Z}_{KN})$ is the output vector.

The computational complexity, under direct computation approach, associated with the SIMO problem is bounded by $\mathbf{O}((KN)^3 + 2(KN)^3) = \mathbf{O}((KN)^3 + 2(KN)^3)$, and it is given by the complexity of pseudo-inverse operator acting over the matrix \mathbf{Z} of order $KN \times KN$, and the complexity of the matrix-vector product of the vector h of length KN . Therefore, the dominant complexity is $\mathbf{O}((KN)^3)$, where K is the number of samples in the impulse response $h(t)$, and N is the number of receiver transducers.

6.2.4 MIMO Case

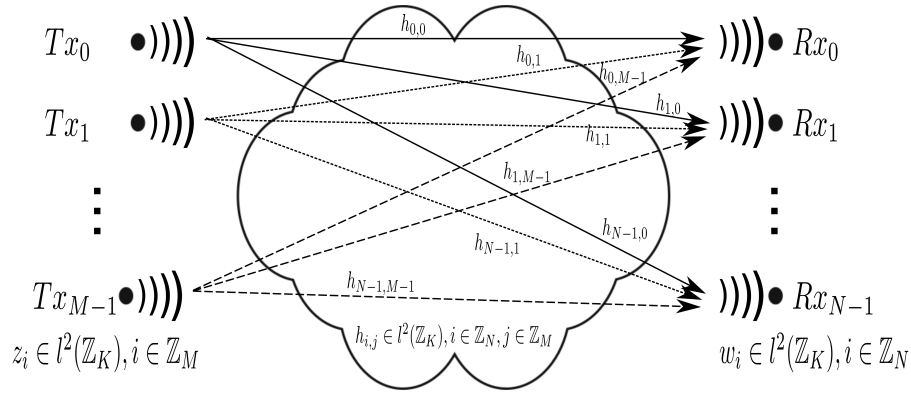


Figure 6-4: Schematic of MIMO Channel System

MIMO case has M transmitter transducers, and N receiver transducers. Equation (6.10) shows the input-output relation in the MIMO channel system.

$$\begin{aligned} \mathbf{w} &= \left[\bigsqcup_{i \in \mathbb{Z}_M} \left(\bigoplus_{j \in \mathbb{Z}_N} \mathcal{C}\{z_i\} \right) \right] \left[\bigvee_{i \in \mathbb{Z}_N} \left(\bigvee_{j \in \mathbb{Z}_M} h_{i,j} \right) \right] = \\ & \left[\bigsqcup_{i \in \mathbb{Z}_M} (I_N \otimes \mathcal{C}\{z_i\}) \right] \left[\bigvee_{i \in \mathbb{Z}_N} \left(\bigvee_{j \in \mathbb{Z}_M} h_{i,j} \right) \right] = \mathbf{Z}\mathbf{h}, \end{aligned} \quad (6.10)$$

where $z_i \in l^2(\mathbb{Z}_K), i \in \mathbb{Z}_M$ is the i -th transmitted signal, $h_{i,j} \in l^2(\mathbb{Z}_K), i \in \mathbb{Z}_N, j \in \mathbb{Z}_M$ is the ij -th *impulse response*, $w_i \in l^2(\mathbb{Z}_K), i \in \mathbb{Z}_N$ is the i -th received signal, \otimes_K is the cyclic convolution operator of order K , \vee is the vertical matrix concatenation operator, \sqcup is the horizontal matrix concatenation operator, $\mathbf{Z} \in l^2(\mathbb{Z}_{KN} \times \mathbb{Z}_{KMN})$ is the input matrix, $\mathbf{h} \in l^2(\mathbb{Z}_{KMN})$ is the impulse response channel vector, $\mathbf{w} \in l^2(\mathbb{Z}_{KN})$ is the output vector, and \mathcal{C} is the circulant operator.

Figure 6–4 illustrates the MIMO system. A matrix-vector representation is shown in Equation (6.11).

$$\begin{bmatrix} w_0[0] \\ \vdots \\ w_0[L-1] \\ \vdots \\ w_{N-1}[0] \\ \vdots \\ w_{N-1}[L-1] \end{bmatrix} = \begin{bmatrix} \mathcal{C}\{z_0\} & \dots & \mathbf{0}_{K \times K} & \dots & \mathcal{C}\{z_{M-1}\} & \dots & \mathbf{0}_{K \times K} \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ \mathbf{0}_{K \times K} & \dots & \mathcal{C}\{z_0\} & \dots & \mathbf{0}_{K \times K} & \dots & \mathcal{C}\{z_{M-1}\} \end{bmatrix} \begin{bmatrix} h_{0,0}[0] \\ \vdots \\ h_{0,0}[L-1] \\ \vdots \\ h_{0,M-1}[0] \\ \vdots \\ h_{0,M-1}[L-1] \\ \vdots \\ h_{N-1,0}[0] \\ \vdots \\ h_{N-1,0}[L-1] \\ \vdots \\ h_{N-1,M-1}[0] \\ \vdots \\ h_{N-1,M-1}[L-1] \end{bmatrix}. \quad (6.11)$$

In training mode, some series of predefined pulses $z_i, i \in \mathbb{Z}_M$ are transmitted. These pulses are compared with the received signals $w_j, j \in \mathbb{Z}_N$ for characterizing

the **MIMO** channel using the inverse relation showed in Equation (6.12).

$$\mathbf{h} = \mathbf{Z}^\dagger \mathbf{w}, \quad (6.12)$$

where $\mathbf{Z} \in l^2(\mathbb{Z}_{KN} \times \mathbb{Z}_{KMN})$ is the input matrix, $\mathbf{h} \in l^2(\mathbb{Z}_{KMN})$ is the impulse response channel vector, and $\mathbf{w} \in l^2(\mathbb{Z}_{KN})$ is the output vector.

The computational complexity, under direct computation approach, associated with the MIMO problem is bounded by $\mathbf{O}((KMN)^3 + 2(K^3N^3M)) = \mathbf{O}((KMN)^3)$, and it is given by the complexity of pseudo-inverse operator acting over the \mathbf{Z} matrix of dimension $KN \times KMN$, and the complexity of the matrix-vector product of the vector w of length KMN . Therefore, the dominant complexity is $\mathbf{O}((KMN)^3)$, where K is the number of samples in the impulse response $h(t)$, M is the number of transmitter transducers, and N is the number of receiver transducers. This complexity may seem small, however, if we consider some typical values (in an average case) to $M = 16, N = 16, K = 16, 536$ then it is possible to obtain a number of floating point operations estimated, using complex variables, of 1.5×10^{20} floating point operations. This number of floating point operations must be carried out in a fraction of the coherence time T_C . The coherent time T_C in shallow water scenarios is on the order of tenths of milliseconds. So, MIMO estimation problem is very hard, and other computational approaches are required. This thesis work explores the use of *matching pursuit* greedy algorithm for addressing the MIMO channel estimation problem.

6.3 Delay-Doppler Estimation Approaches

6.3.1 Delay-Doppler SISO Approach

The first channel parameter estimation problem is addressed under the Single-Input Single-Output (SISO) assumption; it is, considering just one transmitter transducer, and one receiver transducer. In this scenario, a determined number of scatterers L are present between the transmitter and receiver transducers. These scatterers are considered *point targets scatterers* (in opposition to extended targets scatterers) in this thesis work.

Other important aspect to consider is the eventual movement of the scatterers and/or the transmitter and receiver transducers. This movement introduces a new effect on the model. This effect is called *Doppler effect*, and it is expressed as frequency shifts acting over each input signal $z(t)$. These frequency shifts are associated with each scatterer. If the ratio between the bandwidth of the transmitted signal $z(t)$ and the propagation velocity of signals on the medium c is very small, this effect is negligible. However, due to the velocity of sound on the water, the Doppler effect is really significant in this scenario.

Under SISO assumption, we can express the output signal $w(t)$ as the sum of L copies of signal $z(t)$ with time-delay ξ_l , and with frequency-shift (Doppler effect) ν_l , and scaled by respective attenuation factors α_l . Therefore,

$$w(t) = \sum_{l \in \mathbb{Z}_L} \alpha_l z(t - \xi_l) e^{j2\pi\nu_l t} + n(t),$$

$$t \in \mathbb{R}, w, z \in l^2(\mathbb{R}), \quad (6.13)$$

where $\alpha_l \in \mathbb{C}$ are the attenuation factors for each input signal, $\xi_l, \nu_l \in \mathbb{R}$ are the time-delays and frequency-shifts associated with each scatterer l (ξ_0 and ν_0 can be considered zeros), and $n(t)$ is a real-valued independent wide sense stationary (WSS) Gaussian stochastic process that represents noise.

This formulation captures the more important aspects of a physical realization of a underwater ALS SISO channel.

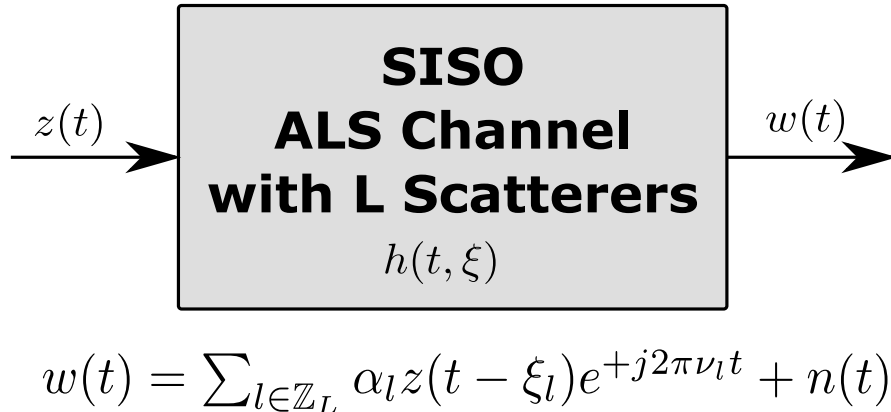


Figure 6–5: ALS Channel under SISO Assumption

Under SISO assumption it is possible to obtain the impulse response of the underwater ALS channel substituting the input signal $z(t)$ by the impulse function $\delta(t)$ in Equation (6.13). Therefore, we obtain

$$h(t, \xi) = \sum_{l=0}^L \alpha_l \delta(t - \xi_l) e^{j2\pi \nu_l t},$$

$$\nu, t, \xi \in \mathbb{R}, l \in \mathbb{Z}_L, h \in l^2(\mathbb{R}), \quad (6.14)$$

where $h(t, \xi)$ (kernel function) is the *time-variant impulse response* of the underwater ALS channel.

It is the most simple case on the underwater ALS model. Impulse response is formulated under the SISO assumption. In the next section, we address the problem of the matrix-vector formulation for ALS channel input-output relation, which establishes the mathematical foundation for the estimation process.

6.3.2 Estimating a Delay-Doppler SISO ALS Channel

A general continuous formulation to SISO ALS channel input-output relationship is shown as follows:

$$w(t) = \int_{\xi \in \mathbb{R}} z(t - \xi)g(t, \xi)d\xi + n(t), \quad (6.15)$$

where $z(t) \in l^2(\mathbb{R})$ is the input signal, $w(t) \in l^2(\mathbb{R})$ is the output signal and $g(t, \xi) \in l^2(\mathbb{R} \times \mathbb{R})$ is the *Input-Delay Spread Function* (used in this thesis work as the *impulse response* of Delay-Doppler ALS channel), and $n(t) \in l^2(\mathbb{R})$ is the noise signal.

In the next stage, the sampling operator is applied to convert the continuous model in a discrete model. This is the first action required to build a computational model. In this case, we call T_S to the sampling time. Now, we choose d to represent the time discrete variable. So, the new formulation is shown as follows

$$w(kT_S) = \sum_{d \in \mathbb{Z}_D} z(kT_S - \xi_d)g(kT_S, \xi_d) + n(kT_S), \quad (6.16)$$

where $k \in \mathbb{Z}_K$, K is the number of samples in the windows, D is the number of delays in the ALS channel, and ξ_d represent equidistant time-delays. We assume that statistically the ALS channel is a WSSUS (wide-sense stationary uncorrelated scattering) channel, and the ξ_{V-1} is very close to the *channel delay spread* T_{Delay} . For estimation purposes, the parameter K must be sufficiently large for offering a good resolution to detect time-delays in the channel.

Making a continuous-to-discrete conversion, we reformulate Equation (6.16) as

$$w[k] = \sum_{d \in \mathbb{Z}_D} z[k - \xi_d]g[k, \xi_d] + n[k], \quad (6.17)$$

where D is the number of time delays.

The time among 2 consecutive delays is denoted as $\Delta\xi$. We considerate that $\Delta\xi = T_S$ for simplicity purposes. This assumption allows to establish that each ξ_d shift corresponds to a $z[k]$ shift. In any case, $\Delta\xi$ must be greater or equal to T_S , and

$\Delta\xi$ must be a multiple of T_S preferably. The principal purpose to establish these restrictions is to reach more mathematical simplicity, and reduce the complexity derived by scaling processes. Other important assumption is that *time spread* T_{Delay} must be less or equal to *coherence time* T_C .

In this moment, we must remember that *Input-Delay Spread Function* $g(t, \xi)$ stays constant during *coherence time* T_C , and so, it stays constant during *time spread* T_{Delay} . Therefore, $g[k, \xi_d]$ could be considered as $g[k]$ in the equation (6.17). Under operators theory perspective, the operation $z[k - \xi_d]$ in (6.17) can be rewritten as $z[k - \xi_d] \rightarrow \mathcal{S}_{\xi_d}\{z[k]\}$, where \mathcal{S} is the *shift operator*, and ξ_d is used as parameter of displacement. So, Equation (6.17) can be reformulate as:

$$w[k] = \sum_{d \in \mathbb{Z}_D} \mathcal{S}_{\xi_d}\{z[k]\}g[k, \xi_d] + n[k]. \quad (6.18)$$

Each linear operator leads to a matrix-vector representation. $\mathcal{S}_{\xi_d}\{z[k]\}$ can be formulated in matrix-vector representation as $(\mathbf{S}_{\xi_d}z)[k]$, where \mathbf{S}_{ξ_d} is the matrix representation of the *shift operator* \mathcal{S}_{ξ_d} . Substituting $(\mathbf{S}_{\xi_d}z)[k]$ by the vector $\mathbf{c}_{\xi_d}[k]$, and remembering that *Input-Delay Spread Function* $g[k, \xi_d]$ stays constant during *time spread* T_{Delay} we can reformulate Equation (6.18) as:

$$w[k] = \sum_{d \in \mathbb{Z}_D} \mathbf{c}_{\xi_d}[k]g[d] + n[k]. \quad (6.19)$$

Making a full matrix-vector representation we obtain:

$$\mathbf{w} = \mathbf{C}\mathbf{g} + \mathbf{n}, \quad (6.20)$$

where \mathbf{w} , and \mathbf{n} are the $K \times 1$ vector representations of $w[k]$, and $n[k]$ respectively, \mathbf{g} is a vector of order D representing $g[d]$, and \mathbf{C} is the $K \times D$ compact matrix representation of $\mathbf{c}_{\xi_d}[k]$.

ALS channels are doubly dispersive channels. Therefore, these channels suffer from delay-Doppler spreading. Therefore, the main problem associated with the

Input-Delay Spread Function $g(t, \xi)$ formulation, when it is used as an estimation tool, is the omission of the Doppler effect. In this case, we use a *Surrogate Function* instead of $g(t, \xi)$. This surrogate function is the *Delay-Doppler Spread Function* $U(\xi, \nu)$, which considers both, delay and Doppler spreading. Continuous *Delay-Doppler Spread Function* $U(\xi, \nu)$ was defined in Equation (5.14) as follows:

$$U(\xi, \nu) = \int_{t \in \mathbb{R}} g(t, \xi) e^{-j2\pi\nu t} dt. \quad (6.21)$$

Under operators theory perspective, we can express Equation (6.21) as follows:

$$U(\xi, \nu) = \mathcal{F}_t\{g(t, \xi)\}, \quad (6.22)$$

where \mathcal{F}_t is the Fourier operator with respect to variable t . Applying the inverse relation we obtain the follow expression:

$$g(t, \xi) = \mathcal{F}_\nu^{-1}\{U(\xi, \nu)\}, \quad (6.23)$$

where \mathcal{F}_ν^{-1} is the inverse Fourier operator with respect to variable ν . Fourier operator admits matrix representation; therefore, we can express, using a full matrix-vector representation, Equation (6.23) as follows:

$$\mathbf{g} = \left(\mathbf{I}_D \otimes \frac{1}{L} \mathbf{F}_L^* \right) \times \mathcal{E}_{DL,1}\{\mathbf{U}\}, \quad (6.24)$$

where \mathbf{g} is the *Input-Delay Spread Function* expressed in vector representation, \mathbf{I}_D is the identity matrix of order D , \otimes is the Kronecker product operator, \mathbf{F}_L is the Fourier matrix of order L , D and L are the number of time-delays and Doppler-shifts respectively, the structure $\frac{1}{L} \mathbf{F}_L^*$ represents the inverse Fourier matrix of order L , and $\mathcal{E}_{r,c}$ is the matrix reshape operator with row parameter r and column parameter c .

In this formulation, the order of \mathbf{g} vector is $D \times L$, and it can be substituted in the equation (6.20) to obtain a new ALS SISO channel input-output relation, using the *Delay-Doppler Spread Function* $U(\xi, \nu)$ instead of the *Input-Delay Spread*

Function $g(t, \xi)$ as channel characterization function. This new relation is shown as follows:

$$\mathbf{w} = \left[\left(\bigvee_{i \in \mathbb{Z}_V} \bigsqcup_{j \in \mathbb{Z}_L} (\phi_{i,j} \mathcal{Y}_{i,D}\{z\}) \right) \times \left(\mathbf{I}_D \otimes \frac{1}{L} \mathbf{F}_L^* \right) \right] \times \mathcal{E}_{DL,1}\{\mathbf{U}\} + \mathbf{n}, \quad (6.25)$$

where V is the length of the sample windows, $z \in l^2(\mathbb{Z}_V)$ is the transmitted signal, $\left(\bigvee_{i \in \mathbb{Z}_V} \bigsqcup_{j \in \mathbb{Z}_L} (\phi_{i,j} \mathcal{Y}_{i,D}\{z\}) \right)$ is a matrix with size $V \times DL$ associated with the new characterization function $U(\xi, \nu)$ used, and \mathcal{Y} is the Windows-Delay-Reverse operator.

Now we define:

$$\mathbf{X} = \left(\bigvee_{i \in \mathbb{Z}_V} \bigsqcup_{j \in \mathbb{Z}_L} (\phi_{i,j} \mathcal{Y}_{i,D}\{z\}) \right) \times \left(\mathbf{I}_D \otimes \frac{1}{L} \mathbf{F}_L^* \right), \quad (6.26)$$

and

$$\mathbf{h} = \mathcal{E}_{DL,1}\{\mathbf{U}\}. \quad (6.27)$$

Obtaining a more simple ALS SISO channel input-output relation as follows:

$$\mathbf{w} = \mathbf{X}\mathbf{h} + \mathbf{n}, \quad (6.28)$$

where the matrix $\mathbf{X} \in l^2(\mathbb{Z}_V \times \mathbb{Z}_{DL})$ and the vector $\mathbf{h} \in l^2(\mathbb{Z}_{DL})$.

The ALS SISO channel estimation problem now can be expressed as the problem of estimating \mathbf{h} matrix, given the input matrix \mathbf{X} , and the output vector \mathbf{w} . Both, \mathbf{X} and \mathbf{w} are known because these matrices are associated with the input and output signals respectively, and these are known during a training stage. This formulation allows the estimation of both, delay and Doppler parameters at the same time.

6.3.3 Delay-Doppler SISO Estimation Strategy using Matching Pursuit Algorithms

A direct computation approach to the channel estimation problem performs an inverse or pseudo-inverse matrix operation on \mathbf{X} , and then, multiply the result by

\mathbf{w} , as follows:

$$\mathbf{h} = \mathbf{X}^\dagger \mathbf{w}. \quad (6.29)$$

Therefore, the computational complexity, in delay-Doppler SISO case, to estimate \mathbf{h} , using a direct computation is bounded by $\mathbf{O}((DL)^3 + 2(DL)^2V)$ (complexity of matrix pseudo-inverse operation is dominant). However, many complications can occur, such as, quasi-singular matrices, non-square matrices, among other complications. Another problem associated with direct computation approach is its expensive computational complexity. In this case, it is possible to address the problem using others approaches. A very useful approach is *matching pursuit* greedy algorithm, developed by Mallat and Zhang, and used in signal processing problems with time-frequency dictionaries [42].

The main reason for choosing the *matching pursuit* approach is the assumption of sparsity on \mathbf{h} vector. The time-frequency resolution used to estimate the *Delay-Doppler Spread Function* $U(\xi, \nu)$ must be enough to detect most significant Delay-Doppler contributions, but usually generates too many zero-valued positions on the \mathbf{h} vector. Intuitively, it is possible to imagine that only some delay-Doppler positions must have significant values.

The *matching pursuit* greedy algorithm evaluates, using inner products, the columns \mathbf{c} in the input matrix \mathbf{X} for determining which of these, $c_i, i \in \mathbb{Z}_{DL}$, have significant contribution on \mathbf{w} vector. For each chosen column c_i one coefficient λ_i is calculated, and \mathbf{w} is updated by subtracting of the contribution $\lambda_i c_i$. So, in each iteration $\mathbf{w} \leftarrow \mathbf{w} - \lambda_i c_i$ is performed. The algorithm stops when the norm of residual vector \mathbf{h} is less than a threshold value or when the rate of change on residual vector \mathbf{g} is too insignificant. This threshold value is associated with a desired accuracy level. The approximate algorithms offer solutions sufficiently accurate, but reducing the computational cost. This trade-off is applied in the channel estimation problem, and can be considered a contribution in this thesis work.

6.3.4 Matching Pursuit Complexity

The complexity of *matching pursuit algorithm* is related directly with the number of significant coefficients (or non-zero coefficients) found in the delay-Doppler spread function, and the order of the matrix \mathbf{X} . On sparse conditions, the number of significant coefficients may be close to 20 percent of the elements content in the vector \mathbf{h} . This assumption (sparsity condition) simplifies the channel problem estimation. The goal is getting delay-Doppler parameters on the *Delay-Doppler Spread Function* $U(\xi, \nu)$.

In Basic Matching Pursuit and Order-Recursive Least Square Matching Pursuit algorithms, the computational work is measured in two stages:

1. computation of inner products (to measure the norm of each column) between each column of input matrix c_j and the output vector w , and
2. number of iterations to identify the dominant channel components and coefficients' estimation.

Each inner product costs $\mathbf{KD}^2\mathbf{L}^2$, but its cost can be reduced to $\mathbf{O}(D^2L^2)$ when the streaming symbol transmission starts, and then, the inner products can be updated recursively.

In the second stage is more difficult to estimate the computational complexity. On basic Matching Pursuit, if the algorithm converges in I iterations, then, the complexity is bounded by $\mathbf{O}(DLI - I^2)$, and in Order-Recursive Least Square Matching Pursuit case, the computational complexity is bounded by $\mathbf{O}(DLI^2)$ [43].

Total complexity, including both stages, on Basic Matching Pursuit is bounded by $\mathbf{O}(DLI^2)$, and on Order-Recursive Least Square Matching Pursuit is bounded $\mathbf{O}(D^2L^2 + DLI^2)$. Where D is the number of tap delays, L is the number of Doppler shifts, and K is the number of samples in the windows. Table 6–1 summaries the complexities associated with matching pursuit algorithms under MIMO assumption.

MP Variant	Type	Complexity	Loop Invariant
Basic MP	Greedy	$\mathbf{O}(MNDLI^2)$	Maximum Contribution
Orthogonal MP	Greedy	$\mathbf{O}((MNDL)^2 + MNDLI^2)$	Maximum Contribution
Order Recursive LS-MP	Greedy	$\mathbf{O}((MNDL)^2 + MNDLI^2)$	Minimum Residual Norm
Direct Computation	Pseudo Inverse	$\mathbf{O}((KMNDL)^3)$	N/A

Table 6–1: Complexity of Matching Pursuit (MP) Algorithm Variants. K = Windows Length, D = Number of Delays, L = the Doppler Shifts, M = Number of Transmitters, N = Number of Receivers, and I = Algorithm Iterations.

Table 6–2: Sequence of selected columns c_i on MP variants. Matrix \mathbf{X} has 150 columns.

Iter	ORLSMP	Residual Norm	OMP	Residual Norm	BMP	Residual Norm
1	40	88.199	40	88.199	40	88.199
2	18	79.717	18	79.717	18	79.730
3	9	69.868	9	69.868	9	69.900
4	66	60.556	66	60.556	66	60.588
5	90	50.711	90	50.711	90	50.765
6	111	41.259	111	41.259	111	41.322
7	126	35.948	126	35.948	126	36.053
8	52	31.034	52	31.034	52	31.211
9	99	25.340	99	25.340	99	25.658
10	63	22.672	12	22.676	107	24.292
11	12	19.729	63	19.729	63	23.074
12	37	16.983	37	16.983	37	21.799
13	107	14.127	107	14.127	20	20.707
14	20	11.652	20	11.652	12	19.613
15	84	9.166	84	9.166	84	19.124
16	39	5.942	39	5.942	5	18.921
17	16	3.766	16	3.766	68	18.755
18	17	2.899	17	2.899	125	18.643
19	42	1.722	42	1.722	39	18.548
20	88	0.000	88	0.000	0	0.000
21	87	0.000	27	0.000	0	0.000

Three variants of *matching pursuit algorithm* were implemented in this thesis work, *basic*, *orthogonal*, and *order-recursive least-square matching pursuit*. *Basic matching pursuit* (BMP), is the most simple implementation on this algorithm. BMP chooses the maximum projection (inner product) of columns set of \mathbf{X} on the output vector w , and extract the correspondent contribution of each maximum on w . *Orthogonal matching pursuit*, adds a new stage, after each iteration it recalculates all the λ_i coefficients associate with each chosen c_i column to force the orthogonal condition on \mathbf{w} residual vector on the subspace spanned by c_i selected columns [44], in order to choice of the columns, it is similar to the former algorithm variant. In the third variant (order-recursive least-square MP) the method of choice the columns is updated. Now each column is chosen such that it minimizes the residual value of \mathbf{w} vector. Therefore, each column is chosen considering the previous set of chosen columns for minimizing the \mathbf{w} residual vector, in order to calculate the λ_i coefficients so that it is similar to the former variant [45].

6.3.5 Delay-Doppler SISO Channel Estimation Results

In this thesis, there were three Matlab *matching pursuit* implementations developed and an estimation experiment was designed in which were assigned the following values to the channel model parameters *number of time delays* ($D = 15$), *number of Doppler shifts* ($L = 10$), *sampling time* $T_S = 1/F_S = 1/(20KHz)$, *window length* $V = 512$, *equidistant delay shift* $\Delta\xi = T_S$, *equidistant Doppler shift* $\Delta\nu = (60Hz) * 2 * \pi/L$. Assuming a sparse condition for *Delay-Doppler Spread Function* $\mathbf{U}(\xi, \nu)$ we obtained the results showed in the Table 6–3.

The Table 6–2 shows the sequence of chosen columns in each MP algorithm variant. We can see that the *order recursive least square matching pursuit* (ORLSMP) algorithm always chooses the column that minimizes the residual norm. This fact is easy to see in 10th iteration. ORLSMP choose the column c_{63} but *orthogonal matching pursuit* (OMP) chose the column c_{12} . However, the former reduced the residual

Table 6–3: Estimation of Delay-Doppler Spread Function using Matching Pursuit

Data Given		Estimated Delay-Doppler Spread Functions		
Pos U	Original U	ORLSMP U	OMP U	BMP U
:	:	:	:	:
5	-	-	-	-0.2029
:	:	:	:	:
9	1.7469	1.7469	1.7469	2.8265
:	:	:	:	:
12	1.3488	1.3488	1.3488	0.4907
:	:	:	:	:
16	0.4847	0.4847	0.4847	-
17	0.3369	0.3369	0.3369	-
18	1.7222	1.7222	1.7222	2.7486
:	:	:	:	:
20	0.6533	0.6533	0.6533	0.4663
:	:	:	:	:
37	0.6503	0.6503	0.6503	0.5542
:	:	:	:	:
39	0.8003	0.8003	0.8003	0.1377
40	1.5270	1.5270	1.5270	2.7703
:	:	:	:	:
42	0.3281	0.3281	0.3281	-
:	:	:	:	:
52	1.3766	1.3766	1.3766	1.3235
:	:	:	:	:
63	1.2866	1.2866	1.2866	0.5500
:	:	:	:	:
66	1.6427	1.6427	1.6427	2.5547
:	:	:	:	:
68	-	-	-	-0.1739
:	:	:	:	:
84	0.7571	0.7571	0.7571	0.3184
:	:	:	:	:
87	-	0.0000	-	-
88	0.2238	0.2238	0.2238	-
:	:	:	:	:
90	1.7190	1.7190	1.7190	2.4535
:	:	:	:	:
99	1.4264	1.4264	1.4264	1.3103
:	:	:	:	:
107	1.1500	1.1500	1.1500	0.5982
:	:	:	:	:
111	1.4823	1.4823	1.4823	2.1579
:	:	:	:	:
125	-	-	-	0.0801
126	1.5073	1.5073	1.5073	1.4781
:	:	:	:	:

norm to 22.672 and the latter reduced the residual norm to 22.676 (4 thousandths plus), despite that the contribution of column c_{12} was greater than the contribution of column c_{63} . *Basic matching pursuit* algorithm was not developed to minimize the residual norm, so it only verifies maximum contribution in each iteration when choosing new columns.

Figure 6–6 shows the *Delay-Doppler Spread Function* estimated using *matching pursuit greedy algorithm* variants.

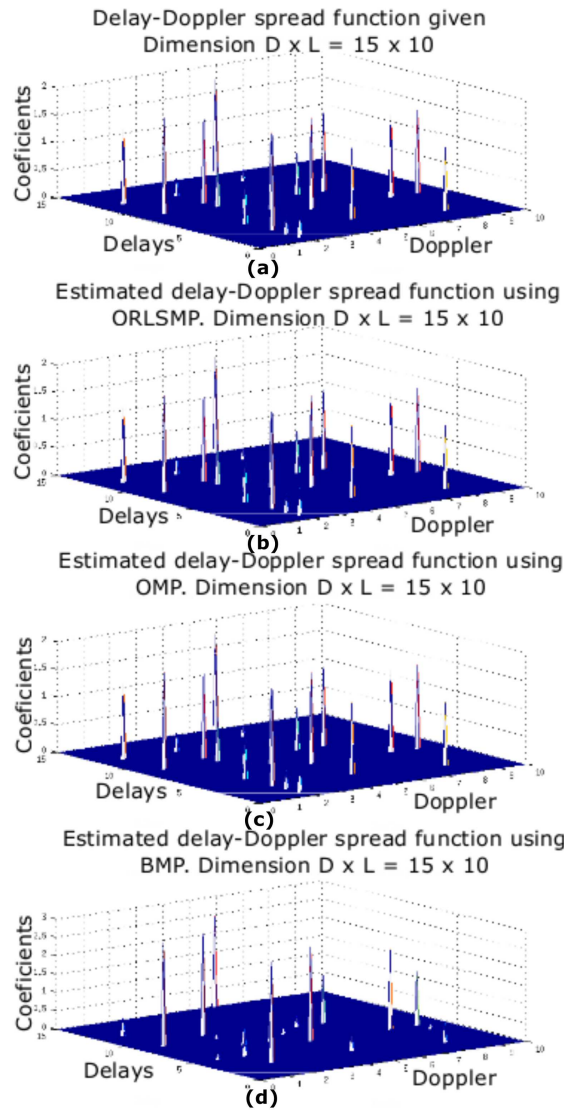


Figure 6–6: Delay-Doppler Spread Function estimated using Matching Pursuit Algorithms: (a) U Given (b) U Est. via ORLSMP (c) U Est. via OMP (d) U Est. via BMP

6.3.6 Delay-Doppler MIMO Approach

This approach of channel parameter estimation is done under the Multiple-Input Multiple-Output (MIMO) assumption; then, consider M transmitter transducers and N receiver transducers. In the MIMO scenario it is possible that a determined

number of scatterers L are present between the transmit and receive transducers. As in the SISO case, these scatterers will be considered point targets scatterers.

We assume that scatterers can be in motion. These movements introduce the Doppler effect and it is expressed as frequency shifts acting over each copy of the transmitted signals $z_m(t)$, associated with each scatterer L . If the ratio between the bandwidth of the transmitted signals $z_m(t)$ and the velocity of propagation of the signal in the medium c is very small, this effect is negligible. However, the velocity of sound in the water (approx. 1,500 meters/sec), the Doppler effect is really significant.

Under the MIMO assumption, we can express the received signals $w_n(t)$ as the sum of L copies of signals $z_m(t)$ delayed by ξ_l , shifted by their respective Doppler frequencies ν_l and scaled by their attenuation factors α_l . Therefore,

$$w_n(t) = \sum_{m \in \mathbb{Z}_M} \sum_{l \in \mathbb{Z}_L} \alpha_{l,m,n} z_m(t - \xi_{l,m,n}) e^{+j2\pi\nu_{l,m,n}t} + \mathbf{n}(t),$$

$$t \in \mathbb{R}, w, z \in l^2(\mathbb{R}), \quad (6.30)$$

where $\alpha_{l,m,n} \in \mathbb{C}$ are the attenuation factors for the input signal m received in the transducer n , $\xi_{l,m,n}, \nu_{l,m,n} \in \mathbb{R}$ are the time delays and Doppler delays associate with each scatterer l , transmitter m , and receiver n ($\xi_{0,m,n}$ and $\nu_{0,m,n}$ can be considered zeros, assuming line of sight between the transmitter and the receiver) and $\mathbf{n}(t)$ is real valued independent wide sense stationary Gaussian stochastic process that represent the noise.

This approach captures the more important aspects of a physical realization of the underwater ALS MIMO channels.

MIMO: Multiple-Input Multiple-Output
ALS: Acoustic Linear Stochastic

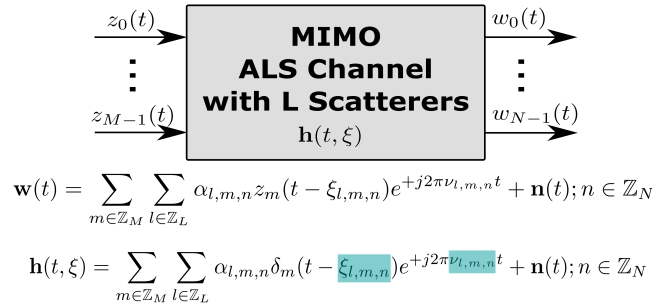


Figure 6–7: ALS Channel under the MIMO assumption

Under the MIMO assumption, we can obtain the impulse response of the underwater ALS channel substituting each input signal $z_m(t)$ by the impulse function $\delta(t)$ in the Equation (6.30). Therefore, we obtain

$$h_{n,m}(t, \xi) = \sum_{m \in \mathbb{Z}_M} \sum_{l=0}^L \alpha_l \delta_{n,m}(t - \xi_{l,m,n}) e^{+j2\pi \nu_{l,m,n} t},$$

$$\nu, t, \xi \in \mathbb{R}, l \in \mathbb{Z}_L, h_{n,m} \in l^2(\mathbb{R}), \quad (6.31)$$

where $h_{n,m}(t, \xi)$ (kernel function) is the *time-variant impulse response* between the transmitter m and the receiver n .

It is the most complex case for the underwater ALS model. In the next section, we will address the problem of the matrix formulation for MIMO ALS channel input-output relationship establishing the mathematical foundations for the estimation process.

6.3.7 Estimating a Delay-Doppler MIMO ALS channel

The more general continuous formulation gives us a MIMO ALS channel input-output relationship as follows:

$$w_n(t) = \sum_{m \in \mathbb{Z}_M} \int_{\xi \in \mathbb{R}} z_m(t - \xi) g(t, \xi) d\xi + \mathbf{n}(t), \quad (6.32)$$

where $z_m(t)$ is the input signal sent by the transmitter m , $w_n(t)$ is the output signal captured by the receiver n and $g_{n,m}(t, \xi)$ is the *Input-Delay Spread Function* (used

in this thesis work as the *impulse response* of Delay-Doppler MIMO ALS channel) between the transmitter n and the receiver n , and $\mathbf{n}(t)$ is the noise signal.

Starting with this formulation, the next step is establishing the sampling relation to convert the continuous model in a discrete model, that is, a basic condition to build a computational model. In this case, we will call T_S the sampling time, then we choose d for the time discrete variable, where the new formulation will be

$$w_n(vT_S) = \sum_{m \in \mathbb{Z}_M} \sum_{d \in \mathbb{Z}_D} z_m(vT_S - \xi_d) g_{n,m}(vT_S, \xi_d) + n(vT_S), \quad (6.33)$$

where $w_n(t)$ is the signal captured by the receiver n , $z_m(t)$ is the signal sent by the transmitter m , $v \in \mathbb{Z}_V$, V is the length of the signal windows, D is the number of delays considered in the ALS channel, and ξ_d are the equidistant time delay. We assume that statistically the ALS channel is a WSSUS (wide-sense stationary uncorrelated scattering) channel, and the ξ_{V-1} is very close to the *channel delay spread* T_{Delay} . For estimation purposes V must be sufficiently large for offering a good resolution for delays detection.

Doing a continuous to discrete conversion, we have rewritten the Equation (6.33) as:

$$w_n[v] = \sum_{m \in \mathbb{Z}_M} \sum_{d \in \mathbb{Z}_D} z_m[v - \xi_d] g_{n,m}[v, \xi_d] + \mathbf{n}[v], \quad (6.34)$$

where D is the number of time delays.

The time distant between 2 consecutive delays will be referenced as $\Delta\xi$ and for simplicity reasons in this thesis work we are considering that $\Delta\xi = T_S$. This condition allows to establish that each ξ_d shift is equivalent to each $z_m[v]$ shift. In any case $\Delta\xi$ could be greater or equal than T_S and preferably $\Delta\xi$ must be an integer multiple of T_S . The principal purpose to establish these restrictions is reach more mathematical simplicity and reduce the complexity of scaling processes. Other important assumption is related to *time spread* T_{Delay} , which must be then less or equal to *coherence time* T_C .

In this point, it must be remembered that the *Input-Delay Spread Function* $g_{n,m}(t, \xi)$, $n \in \mathbb{Z}_N$, $m \in \mathbb{Z}_M$, stays constant during the *coherence time* T_C and so it stays constant during the *time spread* T_{Delay} , therefore, $g_{n,m}[v, \xi_d]$ could be considered as $g_{n,m}[v]$ (independent of the delay) in the Equation (6.34).

Under the operators theory perspective, the operation $z_m[v - \xi_d]$ in (6.17) can be rewritten as $z_m[v - \xi_d] \rightarrow \mathcal{S}_{\xi_d}\{z_m[v]\}$, where \mathcal{S} is the *shift operator* and ξ_d is used as displacement parameter, therefore the Equation (6.34) can be rewritten as:

$$w_n[v] = \sum_{m \in \mathbb{Z}_M} \sum_{d \in \mathbb{Z}_D} \mathcal{S}_{\xi_d}\{z_m[v]\} g_{n,m}[v, \xi_d] + \mathbf{n}[v]. \quad (6.35)$$

Each linear operator leads to a matrix representation, $\mathcal{S}_{\xi_d}\{z_m[v]\}$ can be formulated in matrix-vector representation as $(\mathbf{S}_{\xi_d} z_m)[v]$, where \mathbf{S}_{ξ_d} is the matrix representation of the *shift operator* \mathcal{S}_{ξ_d} . Substituting $(\mathbf{S}_{\xi_d} z_m)[v]$ by the vector $\mathbf{c}_{\xi_d, m}[v]$, and remembering that *Input-Delay Spread Function* $g_{n,m}[v, \xi_d]$ stays constant during *time spread* T_{Delay} we can rewrite the Equation (6.35) as:

$$w_n[v] = \sum_{m \in \mathbb{Z}_M} \sum_{d \in \mathbb{Z}_D} \mathbf{c}_{\xi_d, m}[v] g_{n,m}[d] + \mathbf{n}[v]. \quad (6.36)$$

Applying a full matrix-vector representation we will obtain

$$\mathbf{w}_n = \mathbf{C}_m \mathbf{g}_{n,m} + \mathbf{n}, \quad (6.37)$$

where \mathbf{w}_n , and \mathbf{n} are the vectors of order V , $\mathbf{g}_{n,m}$ is a vector of order D representing $g_{n,m}[d]$, and \mathbf{C}_m is the $V \times D$ matrix representation of $\mathbf{c}_{\xi_d, m}[v]$.

The ALS channels are doubly dispersive channels, then, these channels suffer of delay-Doppler spreading, therefore the main deficiency associates with the *Input-Delay Spread Function* $g_{n,m}(t, \xi)$, when it is used as estimation tool, it is the resulting omission of the Doppler effect. In this situation is appropriate to use a *surrogate function* of the $g_{n,m}(t, \xi)$, that is the *Delay-Doppler Spread Function* $U_{n,m}(\xi, \nu)$, $n \in \mathbb{Z}_N$, $m \in \mathbb{Z}_M$, which considers both, delay and Doppler spreads. The continuous

Delay-Doppler Spread Function $U_{n,m}(\xi, \nu)$ was defined in the Equation (5.14) as:

$$U_{n,m}(\xi, \nu) = \int_{t \in \mathbb{R}} g_{n,m}(t, \xi) e^{-j2\pi\nu t} dt, n \in \mathbb{Z}_N, m \in \mathbb{Z}_M. \quad (6.38)$$

Again, under the operators theory perspective, we can express the Equation (6.38) as:

$$U_{n,m}(\xi, \nu) = \mathcal{F}_t\{g_{n,m}(t, \xi)\}, n \in \mathbb{Z}_N, m \in \mathbb{Z}_M, \quad (6.39)$$

where \mathcal{F}_t is the Fourier operator with respect to t variable, and applying the inverse relation we can obtain

$$g_{n,m}(t, \xi) = \mathcal{F}_\nu^{-1}\{U_{n,m}(\xi, \nu)\}, n \in \mathbb{Z}_N, m \in \mathbb{Z}_M, \quad (6.40)$$

where \mathcal{F}_ν^{-1} is the inverse Fourier operator with respect to ν variable. Fourier operator admits matrix representation, therefore we can express, in discrete manner and using the full matrix-vector representation, the Equation (6.40) as:

$$\mathbf{g}_{n,m} = \left(\mathbf{I}_D \otimes \frac{1}{L} \mathbf{F}_L^* \right) \times \mathcal{E}_{DL,1}\{\mathbf{U}_{n,m}\}, n \in \mathbb{Z}_N, m \in \mathbb{Z}_M, \quad (6.41)$$

where $\mathbf{g}_{n,m}$ is the *Input-Delay Spread Function* between the transmitter m and the receiver n expressed in vectorial manner, \mathbf{I}_D is the identity matrix of order D , \otimes is the Kronecker product operator, \mathbf{F}_L is the Fourier matrix of size L , D and L are the number of delay and Doppler shifts respectively, the structure $\frac{1}{L} \mathbf{F}_L^*$ represents the inverse Fourier matrix of size L , and $\mathcal{E}_{r,c}$ is the matrix reshape operator with row parameter r and column parameter c .

In this formulation, the order of $\mathbf{g}_{n,m}$ vector is DL and it can be substituted in the Equation (6.37) for obtaining a new ALS MIMO channel input-output relationship using the *Delay-Doppler Spread Function* $U_{n,m}(\xi, \nu)$ instead of the *Input-Delay Spread Function* $g_{n,m}(t, \xi)$ as channel characterization function. This new relation

is as follows

$$\mathbf{w}_n = \left[\left(\bigvee_{i \in \mathbb{Z}_V} \bigsqcup_{j \in \mathbb{Z}_L} (\phi_{i,j} \mathcal{Y}_{i,D}\{z_m\}) \right) \times \left(\mathbf{I}_D \otimes \frac{1}{L} \mathbf{F}_L^* \right) \right] \times \mathcal{E}_{DL,1}\{\mathbf{U}_{n,m}\} + \mathbf{n}, \quad (6.42)$$

where $z_m \in l^2(\mathbb{Z}_V)$ is the input signal sent by the transmitter m , w_n is the output signal captured by receiver n , $\left(\bigvee_{i \in \mathbb{Z}_V} \bigsqcup_{j \in \mathbb{Z}_L} (\phi_{i,j} \mathcal{Y}_{i,D}\{z_m\}) \right)$ is a matrix with size $V \times DL$ due to the new function of channel characterization $U_{n,m}(\xi, \nu)$ used. Now we define

$$\mathbf{X}_m = \left(\bigvee_{i \in \mathbb{Z}_V} \bigsqcup_{j \in \mathbb{Z}_L} (\phi_{i,j} \mathcal{Y}_{i,D}\{z_m\}) \right) \times \left(\mathbf{I}_D \otimes \frac{1}{L} \mathbf{F}_L^* \right), n \in \mathbb{Z}_N, m \in \mathbb{Z}_M, \quad (6.43)$$

and

$$\mathbf{h}_{n,m} = \mathcal{E}_{DL,1}\{\mathbf{U}_{n,m}\}, n \in \mathbb{Z}_N, m \in \mathbb{Z}_M, \quad (6.44)$$

obtaining an ALS MIMO channel input-output relationship as follows:

$$\mathbf{w}_n = \mathbf{X}_m \mathbf{h}_{n,m} + \mathbf{n}, n \in \mathbb{Z}_N, m \in \mathbb{Z}_M, \quad (6.45)$$

where each matrix $\mathbf{X}_m \in l^2(\mathbb{Z}_V \times \mathbb{Z}_{DL})$, and each vector $\mathbf{h}_{n,m} \in l^2(\mathbb{Z}_{DL})$.

The ALS MIMO channel estimation problem now can be expressed as the estimate problem of estimating $\mathbf{h}_{n,m}$ matrices given the input matrices \mathbf{X}_m and the output vectors \mathbf{w}_n . Both, \mathbf{X}_m and \mathbf{w}_n can be known because these are associated with input and output signals respectively, and these can be used during a training stage. The relevance of this formulation is that the channel vector $\mathbf{h}_{n,m}$ allows to estimate both, delay and Doppler parameters at the same time.

6.3.8 Delay-Doppler MIMO Estimation Strategy

We start this section remembering the Equation (6.11) which establishes the input-output relation in a MIMO case:

$$\begin{bmatrix} w_0[0] \\ \vdots \\ w_0[L-1] \\ \vdots \\ w_{N-1}[0] \\ \vdots \\ w_{N-1}[L-1] \end{bmatrix} = \begin{bmatrix} \mathcal{C}\{z_0\} & \dots & \mathbf{0}_{K \times K} & \dots & \mathcal{C}\{z_{M-1}\} & \dots & \mathbf{0}_{K \times K} \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ \mathbf{0}_{K \times K} & \dots & \mathcal{C}\{z_0\} & \dots & \mathbf{0}_{K \times K} & \dots & \mathcal{C}\{z_{M-1}\} \end{bmatrix} \begin{bmatrix} h_{0,0}[0] \\ \vdots \\ h_{0,0}[L-1] \\ \vdots \\ h_{0,M-1}[0] \\ \vdots \\ h_{0,M-1}[L-1] \\ \vdots \\ h_{N-1,0}[0] \\ \vdots \\ h_{N-1,0}[L-1] \\ \vdots \\ h_{N-1,M-1}[0] \\ \vdots \\ h_{N-1,M-1}[L-1] \end{bmatrix}. \quad (6.46)$$

Now we consider a new formulation for the delay-Doppler MIMO case starting with the general MIMO case formulation. In the delay-Doppler MIMO scenario the matrix $\mathcal{C}\{z_m\}$, $m \in \mathbb{Z}_M$ used in the general MIMO case is equivalent to the matrix X_m , $m \in \mathbb{Z}_M$ used in the delay-Doppler MIMO formulation showed in the Equation (6.45). Therefore, the Equation (6.11) can be rewritten in the delay-Doppler MIMO

context as follows:

$$\begin{bmatrix} \mathbf{w}_0[0] \\ \vdots \\ \mathbf{w}_0[V-1] \\ \vdots \\ \mathbf{w}_{N-1}[0] \\ \vdots \\ \mathbf{w}_{N-1}[V-1] \end{bmatrix} = \begin{bmatrix} \mathbf{X}_0 & \dots & \mathbf{0}_{V \times DL} & \dots & \mathbf{X}_{M-1} & \dots & \mathbf{0}_{V \times DL} \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ \mathbf{0}_{V \times DL} & \dots & \mathbf{X}_0 & \dots & \mathbf{0}_{V \times DL} & \dots & \mathbf{X}_{M-1} \end{bmatrix} \begin{bmatrix} \mathbf{h}_{0,0}[0] \\ \vdots \\ \mathbf{h}_{0,0}[KL-1] \\ \vdots \\ \mathbf{h}_{0,M-1}[0] \\ \vdots \\ \mathbf{h}_{0,M-1}[KL-1] \\ \vdots \\ \mathbf{h}_{N-1,0}[0] \\ \vdots \\ \mathbf{h}_{N-1,0}[KL-1] \\ \vdots \\ \mathbf{h}_{N-1,M-1}[0] \\ \vdots \\ \mathbf{h}_{N-1,M-1}[KL-1] \end{bmatrix}, \quad (6.47)$$

where M is the number of transmitters, N is the number of receivers, V is the length of the signal window, D , and L is the number of time delays and the number of Doppler shifts considered in each impulse response, $\mathbf{X}_i \in l^2(\mathbb{Z}_V \times \mathbb{Z}_{DL})$, $i \in \mathbb{Z}_M$. In compact matrix-vector representation we can express the Equation (6.47) as follows:

$$\mathbf{W}_{MIMO} = \mathbf{X}_{MIMO} \mathbf{U}_{MIMO}, \quad (6.48)$$

where $\mathbf{W}_{MIMO} \in l^2(\mathbb{Z}_{NV})$, $\mathbf{X}_{MIMO} \in l^2(\mathbb{Z}_{NV} \times \mathbb{Z}_{MNDL})$, and $\mathbf{U}_{MIMO} \in l^2(\mathbb{Z}_{MNDL})$. Using a direct computation approach on the delay-Doppler MIMO estimation problem could be reduced to find an inverse or pseudo-inverse matrix for \mathbf{X}_{MIMO} and

then multiply it by \mathbf{W}_{MIMO} , as follows:

$$\mathbf{U}_{MIMO} = \mathbf{X}_{MIMO}^\dagger \mathbf{W}_{MIMO} \quad (6.49)$$

Therefore, the computational complexity in the delay-Doppler MIMO channel to estimate \mathbf{U}_{MIMO} using a direct computation is $\mathbf{O}((MNDL)^3 + 2(N^3(MDL)^2V))$. However, many complications can appear, such as quasi-singular matrices, non-square matrices, and others. Another problem associated with the direct approach is its expensive computational complexity. In this sense, it is possible to address the problem using others approaches. In this thesis, we used *matching pursuit* greedy algorithm again for resolving the delay-Doppler MIMO estimation problem.

In a typical case of delay-Doppler MIMO parameter estimation with values $\{16,16,16536,128,128\}$ assigned to the complex variables $\{M, N, V, D, L\}$, we obtain 1.7×10^{20} floating-point operations, approximately. These results must be carried out in a fraction of the coherence time T_C . This consideration forces us to find other approaches. In this work, the application of the *matching pursuit* greedy algorithm was revised.

The principal reason for choosing the *matching pursuit* approach is the assumption of sparsity of \mathbf{U}_{MIMO} vector. The time-frequency resolution used for estimating the *Delay-Doppler Spread Function* $U(\xi, \nu)$ is enough to detect the more significant delay-Doppler contributions but usually generates too many zero-value positions on the \mathbf{U}_{MIMO} vector. In a typical delay-Doppler channel, only 20% of the values in the matrix \mathbf{U}_{MIMO} are non-zero or significant values.

The computational complexity in *matching pursuit* algorithm is not clearly defined. However, we can assume a significant reduction of the computational complexity reached by the sparsity condition of the vector \mathbf{U}_{MIMO} . We used simulations to measure the improvement offers by the matching pursuit Greedy algorithm. The next section shows the results.

6.3.9 Delay-Doppler MIMO Channel Estimation Results

Three variants of matching pursuit algorithm were developed to estimate the *delay-Doppler Spread function* \mathbf{U} in a 2×2 MIMO case. These variants are: Basic Matching Pursuit (BMP), Orthogonal Matching Pursuit (OMP), and Order Recursive Least Square Matching Pursuit (ORLSMP). These algorithms were codified using Matlab. An estimation experiment was designed assigning the following values to the parameters: *number of time delays* ($D = 15$), *number of Doppler shifts* ($L = 10$), *sampling time* $T_S = 1/F_S = 1/(20\text{KHz})$, *window length* $V = 512$, *equidistant delay shift* $\Delta\xi = T_S$, *equidistant Doppler shift* $\Delta\nu = (60\text{Hz}) * 2 * \pi/L$, *number of transmit transducers* $M = 2$, *number of receive transducers* $N = 2$. Assuming a sparse condition for *Delay-Doppler Spread Function* $\mathbf{U}(\xi, \nu)$ we obtained the results showed in the Tables 6-4, 6-5, 6-6, 6-7.

In this experiment, a random MIMO matrix $\mathbf{X}_{MIMO} \in l^2(\mathbb{Z}_{1024} \times \mathbb{Z}_{600})$ was built with the structure described in the Equations (6.47) and (6.48), and a random sparse matrix $\mathbf{U}_{MIMO} \in \mathbb{Z}_{600}$. Using the Equation (6.48) the channel action was simulated on the input matrix \mathbf{X}_{MIMO} and it was carried out on the output matrix $\mathbf{W}_{MIMO} \in l^2(\mathbb{Z}_{1024})$. The matrices \mathbf{X}_{MIMO} and \mathbf{W}_{MIMO} were passed as parameters to *matching pursuit algorithms*. The purposes of this experiment were: first, to estimate the matrix \mathbf{U}_{MIMO} using matching pursuit algorithm and, second, to measure, in approximated manner, the computational complexity in a sparse MIMO channel condition.

The Tables 6-8 and 6-9 show the sequence of chosen columns in each Matching Pursuit algorithm variant. We can see that *order recursive least square matching pursuit* (ORLSMP) algorithm always chooses the column that minimizes the residual norm. This fact is easy to watch in 14th iteration, ORLSMP chose the column c_{121} but *orthogonal matching pursuit* (OMP) chose the column c_{144} . However, the ORLSMP reduced the residual norm to 93.59 while the OMP only reduced the

Table 6–4: 2×2 MIMO Case. Estimation of delay-Doppler Spread Function using Matching Pursuit Greedy Algorithms. $U_{0,0}$ function. Positions 0-149 in U_{MIMO} .

Data Given		Estimated Delay-Doppler Spread Functions		
Pos $U_{0,0}$	Original $U_{0,0}$	ORLSMP $U_{0,0}$	OMP $U_{0,0}$	BMP $U_{0,0}$
:	:	:	:	:
5	2.3361	2.3361	3.3720	1.2492
6	2.1917	2.1917	0	0
:	:	:	:	:
8	0	0	1.9760	4.7029
:	:	:	:	:
14	1.4348	1.4348	0	0
:	:	:	:	:
26	1.8971	1.8971	3.1259	0
27	2.5533	2.5533	3.5069	9.5730
28	1.9765	1.9765	0	0
:	:	:	:	:
38	2.4847	2.4847	3.2132	7.5689
39	1.2168	1.2168	0	0
:	:	:	:	:
43	2.1559	2.1559	2.6409	0
44	2.8537	2.8537	2.7511	1.2731
:	:	:	:	:
52	1.4224	1.4224	0	0
:	:	:	:	:
68	1.4994	1.4994	0	0
:	:	:	:	:
92	1.3051	1.3051	0	0
:	:	:	:	:
96	1.0039	1.0039	0	0
:	:	:	:	:
115	2.7458	2.7458	2.9249	5.5397
:	:	:	:	:
121	2.5105	2.5105	3.5352	0
:	:	:	:	:
124	2.1486	2.1486	0	0
:	:	:	:	:
127	1.9171	1.9171	3.2537	11.8255
:	:	:	:	:
130	1.0670	1.0670	0	0
:	:	:	:	:
132	2.3152	2.3152	3.1350	0
133	1.6625	1.6625	0	0
134	2.9348	2.9348	4.1011	2.7532
:	:	:	:	:
141	1.0159	1.0159	0	0
:	:	:	:	:
143	1.4778	1.4778	0	0
144	2.2653	2.2653	3.9799	3.7647
:	:	:	:	:

Table 6–5: 2×2 MIMO Case. Estimation of delay-Doppler Spread Function using Matching Pursuit Greedy Algorithms. $\mathbf{U}_{1,0}$ function. Positions 150-299 in \mathbf{U}_{MIMO} .

Data Given		Estimated Delay-Doppler Spread Functions		
Pos $U_{1,0}$	Original $U_{1,0}$	ORLSMP $U_{1,0}$	OMP $U_{1,0}$	BMP $U_{1,0}$
:	:	:	:	:
153	1.6757	1.6757	0	0
:	:	:	:	:
170	1.5413	1.5413	0	0
:	:	:	:	:
183	1.6938	1.6938	0	0
:	:	:	:	:
187	1.3729	1.3729	2.1431	3.2565
188	1.1974	1.1974	1.9197	0
:	:	:	:	:
200	2.2461	2.2461	0	0
201	2.8673	2.8673	3.5858	1.3568
:	:	:	:	:
204	0	0	1.3157	7.4830
:	:	:	:	:
209	2.5833	2.5833	3.6436	0
:	:	:	:	:
228	1.0010	1.0010	0	0
:	:	:	:	:
238	2.7077	2.7077	3.8064	1.2645
:	:	:	:	:
243	2.0051	2.0051	2.5568	0
:	:	:	:	:
248	1.9284	1.9284	2.8804	4.3527
249	1.8269	1.8269	0	0
:	:	:	:	:
259	1.6835	1.6835	0	0.7795
:	:	:	:	:
262	1.0663	1.0663	0	0
:	:	:	:	:
265	1.4256	1.4256	3.4081	2.9871
:	:	:	:	:
286	2.5754	2.5754	3.0392	2.6570
:	:	:	:	:
291	2.6207	2.6207	3.1085	0
:	:	:	:	:
293	1.7061	1.7061	2.2886	11.9775
:	:	:	:	:
298	2.6647	2.6647	3.1884	0

Table 6–6: 2×2 MIMO Case. Estimation of delay-Doppler Spread Function using Matching Pursuit Greedy Algorithms. $\mathbf{U}_{0,1}$ function. Positions 300-449 in \mathbf{U}_{MIMO} .

Data Given		Estimated Delay-Doppler Spread Functions		
Pos $U_{0,1}$	Original $U_{0,1}$	ORLSMP $U_{0,1}$	OMP $U_{0,1}$	BMP $U_{0,1}$
:	:	:	:	:
318	1.9796	1.9796	0	0
:	:	:	:	:
322	2.9443	2.9443	4.4812	2.0434
:	:	:	:	:
353	2.7039	2.7039	3.5407	3.1537
:	:	:	:	:
375	2.6329	2.6329	3.7946	3.8296
376	1.4181	1.4181	0	0
:	:	:	:	:
385	2.9188	2.9188	4.0628	3.2094
:	:	:	:	:
392	1.9998	1.9998	2.9108	0.8845
:	:	:	:	:
401	2.9914	2.9914	4.0493	6.3969
:	:	:	:	:
413	2.7263	2.7263	2.6200	0
:	:	:	:	:

residual norm to 95.595, despite the contribution of column c_{144} was greater than the contribution of column c_{121} . Applying an analysis to tables presented in this section, it easily highlights the fact that the *Order Recursive Least Square Matching Pursuit* (ORLSMP) is the *matching pursuit* variant that offers the best results on the MIMO case. In the SISO case, reviewed above, the advantages offered by OLSRMP over OMP were minimum. However, on the MIMO case, OLSRMP is radically better than OMP. In the MIMO case the *Basic Matching Pursuit* (BMP) algorithm variant is highly erratic. Therefore, this variant is not useful in the MIMO scenarios.

Figures 6–8, 6–9, 6–10, 6–11 show the 2×2 MIMO *Delay-Doppler Spread Functions* estimated using three *matching pursuit greedy algorithm* variants. All these functions were computed using an unique matrix-vector formulation. The integrated approach carried out a solution contents in only one big vector \mathbf{U}_{MIMO} but separated using indexes discrimination. In these figures, we can appreciate that OLSRMP is the *matching pursuit algorithm* more appropriate to MIMO channel scenarios.

Table 6–7: 2×2 MIMO Case. Estimation of delay-Doppler Spread Function using Matching Pursuit Greedy Algorithms. $\mathbf{U}_{1,1}$ function. Positions 450-599 in \mathbf{U}_{MIMO} .

Data Given		Estimated Delay-Doppler Spread Functions		
Pos $U_{1,1}$	Original $U_{1,1}$	ORLSMP $U_{1,1}$	OMP $U_{1,1}$	BMP $U_{1,1}$
:	:	:	:	:
452	1.5825	1.5825	2.6072	8.5423
:	:	:	:	:
455	2.8261	2.8261	3.7564	1.7326 +
:	:	:	:	:
459	1.9381	1.9381	0	0
:	:	:	:	:
463	2.5409	2.5409	3.6116	0
:	:	:	:	:
466	2.7413	2.7413	3.0498	0.9142
:	:	:	:	:
478	2.4262	2.4262	3.2608	5.1936
:	:	:	:	:
482	0	0	0	0.7697
:	:	:	:	:
507	1.2393	1.2393	0	0
:	:	:	:	:
527	2.0883	2.0883	0	0
:	:	:	:	:
536	2.3547	2.3547	3.4629	6.3446
:	:	:	:	:
579	2.7487	2.7487	2.8741	2.4921
:	:	:	:	:
592	2.7664	2.7664	0	0
:	:	:	:	:
599	2.3643	2.3643	2.8363	0

Table 6–8: Sequence of chosen columns c_i in Matching Pursuit algorithm variants. Using a \mathbf{X}_{MIMO} matrix of 600 columns. Part 1. Iterations 1-40.

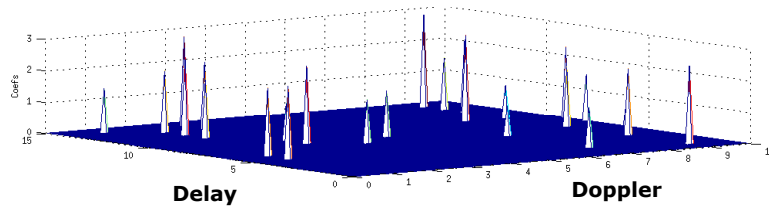
Iter	ORLSMP	Residual Norm	OMP	Residual Norm	BMP	Residual Norm
1	293	188.073	293	188.073	293	188.073
2	127	171.984	127	171.984	127	171.984
3	27	160.644	27	160.644	27	160.650
4	452	150.680	452	150.680	452	150.694
5	38	142.568	38	142.568	38	142.603
6	204	134.174	204	134.174	204	134.240
7	401	127.358	401	127.358	401	127.430
8	536	120.505	536	120.505	536	120.610
9	115	115.134	115	115.134	115	115.255
10	478	110.064	478	110.064	478	110.229
11	8	105.657	8	105.657	8	105.920
12	248	101.797	248	101.797	248	102.134
13	298	97.593	375	98.749	375	99.046
14	121	93.590	144	95.595	144	96.020
15	68	90.335	385	93.096	385	93.651
16	144	86.967	187	90.625	187	91.260
17	455	84.193	353	88.159	353	88.913
18	187	81.190	298	83.171	265	86.789
19	322	78.220	134	78.129	134	84.978
20	385	75.313	265	75.511	286	83.223
21	353	72.358	579	73.462	579	81.629
22	265	69.524	322	69.941	322	80.509
23	209	66.646	455	66.335	455	79.695
24	134	64.003	209	63.484	201	79.205
25	291	61.378	286	60.936	44	78.746
26	44	59.006	44	58.718	5	78.320
27	579	56.765	121	56.305	238	77.901
28	238	54.913	238	54.337	466	77.667
29	5	53.099	5	52.658	392	77.449
30	286	51.513	392	50.991	259	77.246
31	466	49.889	466	49.380	482	77.083
32	463	48.175	201	47.712	0	0.000
33	201	46.292	463	45.919	0	0.000
34	392	44.620	291	44.013	0	0.000
35	243	43.186	243	42.559	0	0.000
36	26	41.791	413	41.139	0	0.000
37	132	40.315	26	39.450	0	0.000
38	375	38.797	132	37.802	0	0.000
39	413	37.197	43	36.414	0	0.000
40	600	35.779	600	34.965	0	0.000

Table 6–9: Sequence of chosen columns c_i in Matching Pursuit algorithm variants. Using a \mathbf{X}_{MIMO} matrix of 600 columns. Part 2. Iterations 41-69.

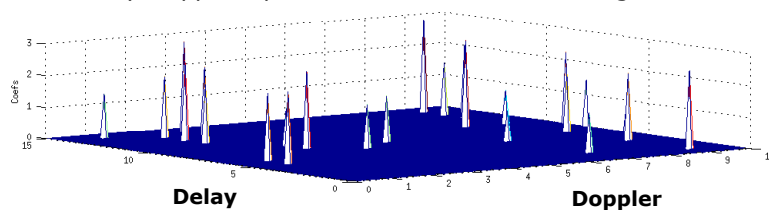
Iter	ORLSMP	Residual Norm	OMP	Residual Norm	BMP	Residual Norm
41	43	34.347	188	33.982	0	0.000
42	200	33.191	0	0.000	0	0.000
43	6	32.097	0	0.000	0	0.000
44	28	30.971	0	0.000	0	0.000
45	124	29.671	0	0.000	0	0.000
46	527	28.482	0	0.000	0	0.000
47	183	27.203	0	0.000	0	0.000
48	459	25.904	0	0.000	0	0.000
49	592	24.528	0	0.000	0	0.000
50	259	23.110	0	0.000	0	0.000
51	143	21.847	0	0.000	0	0.000
52	249	20.737	0	0.000	0	0.000
53	318	19.582	0	0.000	0	0.000
54	52	18.559	0	0.000	0	0.000
55	133	17.447	0	0.000	0	0.000
56	170	16.330	0	0.000	0	0.000
57	92	15.160	0	0.000	0	0.000
58	376	13.917	0	0.000	0	0.000
59	153	12.653	0	0.000	0	0.000
60	14	11.471	0	0.000	0	0.000
61	188	10.450	0	0.000	0	0.000
62	507	9.432	0	0.000	0	0.000
63	39	8.379	0	0.000	0	0.000
64	262	7.344	0	0.000	0	0.000
65	130	6.313	0	0.000	0	0.000
66	141	5.113	0	0.000	0	0.000
67	228	3.561	0	0.000	0	0.000
68	96	0.000	0	0.000	0	0.000
69	155	0.000	0	0.000	0	0.000

$$\mathbf{U}_{0,0} \in l^2(\mathbb{Z}_{15} \times \mathbb{Z}_{10})$$

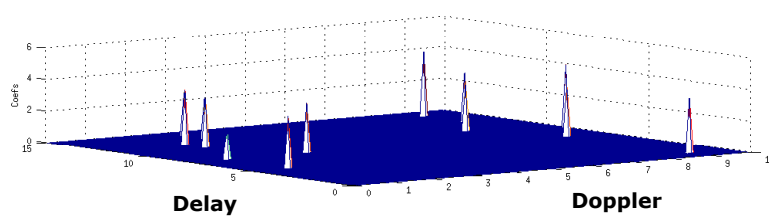
Delay-Doppler Spread Function Given between transmitter 0 and receiver 0.



Delay-Doppler Spread Function Estimated using ORLSMP



Delay-Doppler Spread Function Estimated using OMP



Delay-Doppler Spread Function Estimated using BMP

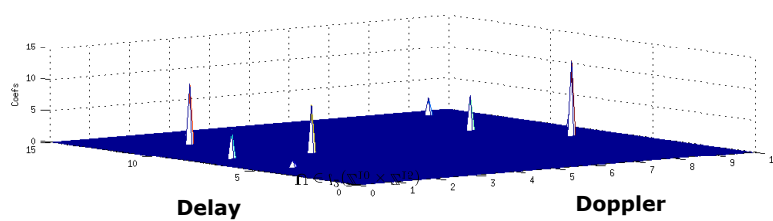
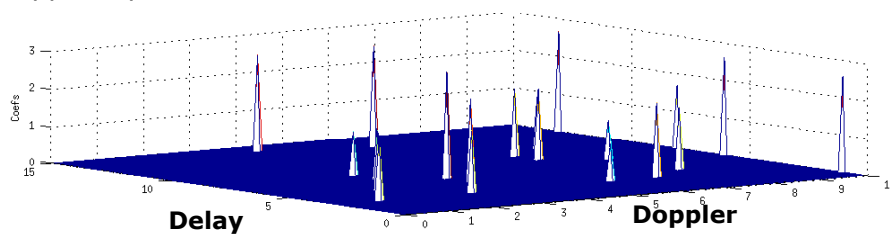


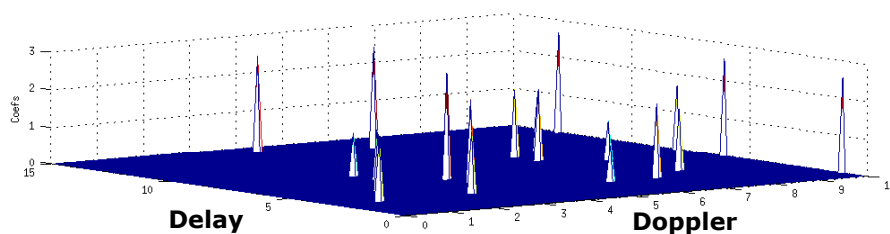
Figure 6–8: 2×2 MIMO case. Delay-Doppler Spread Function $\mathbf{U}_{0,0}$ estimated using Matching Pursuit Algorithms

$$\mathbf{U}_{1,0} \in l^2(\mathbb{Z}_{15} \times \mathbb{Z}_{10})$$

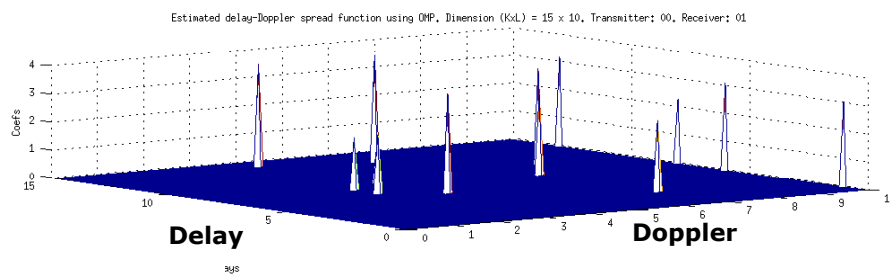
Delay-Doppler Spread Function Given between transmitter 0 and receiver 1



Delay-Doppler Spread Function Estimated using ORLSMP



Delay-Doppler Spread Function Estimated using OMP



Delay-Doppler Spread Function Estimated using BMP

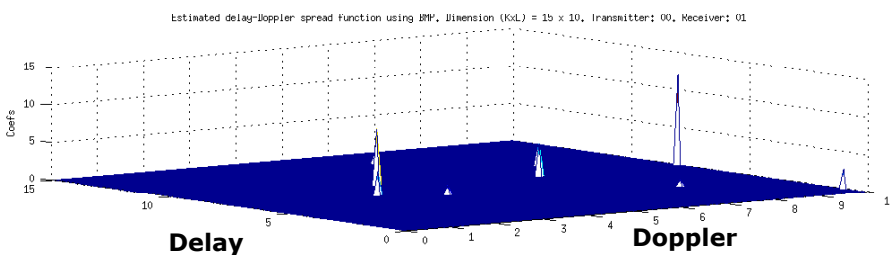
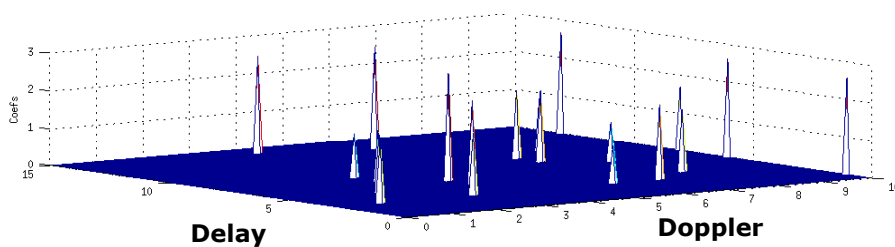


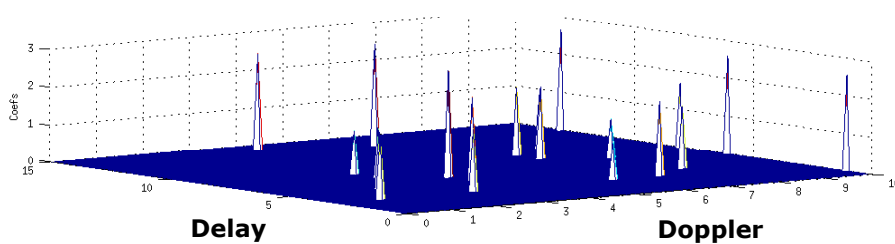
Figure 6–9: 2×2 MIMO case. Delay-Doppler Spread Function $\mathbf{U}_{1,0}$ estimated using Matching Pursuit Algorithms

$$\mathbf{U}_{0,1} \in l^2(\mathbb{Z}_{15} \times \mathbb{Z}_{10})$$

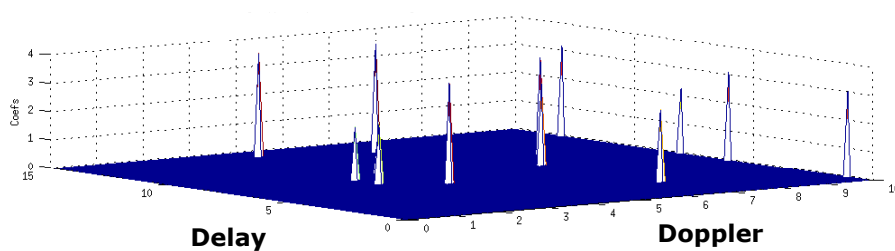
Delay-Doppler Spread Function Given between transmitter 1 and receiver 0



Delay-Doppler Spread Function Estimated using ORLSMP



Delay-Doppler Spread Function Estimated using OMP



Delay-Doppler Spread Function Estimated using BMP

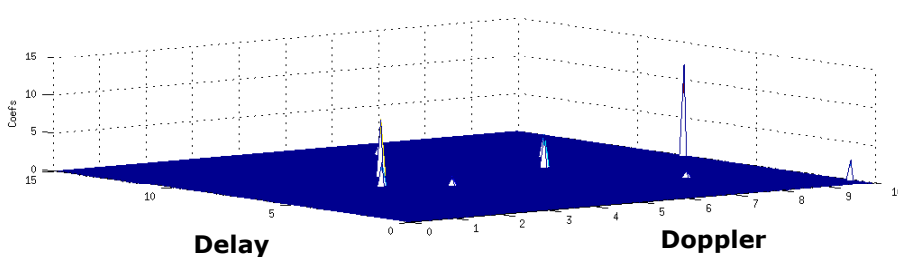
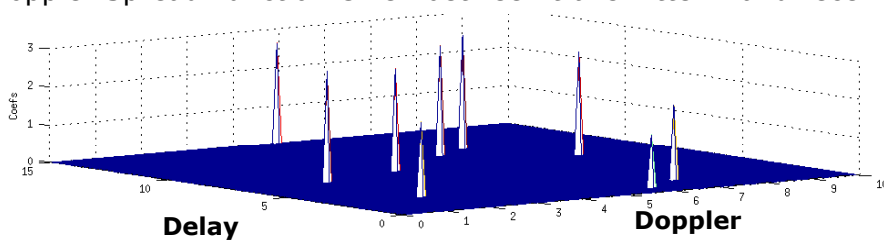


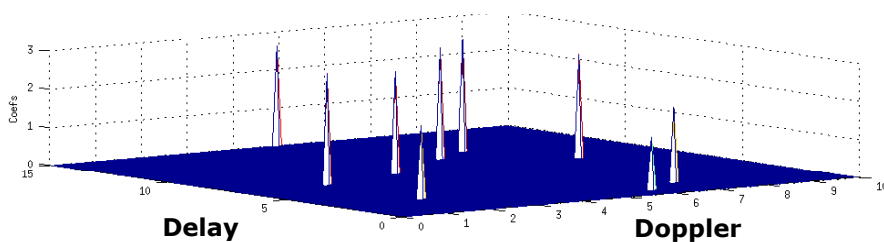
Figure 6–10: 2×2 MIMO case. Delay-Doppler Spread Function $\mathbf{U}_{0,1}$ estimated using Matching Pursuit Algorithms

$$\mathbf{U}_{1,1} \in l^2(\mathbb{Z}_{15} \times \mathbb{Z}_{10})$$

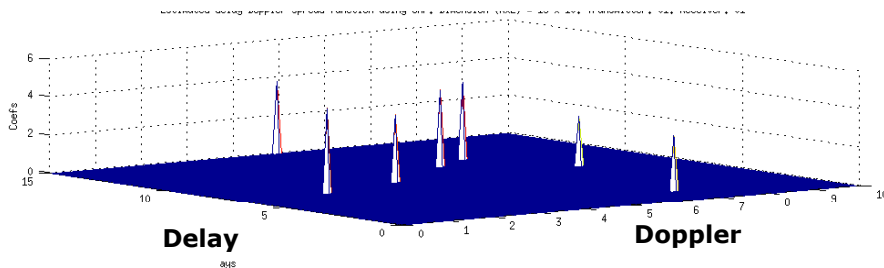
Delay-Doppler Spread Function Given between transmitter 1 and receiver 1



Delay-Doppler Spread Function Estimated using ORLSMP



Delay-Doppler Spread Function Estimated using OMP



Delay-Doppler Spread Function Estimated using BMP

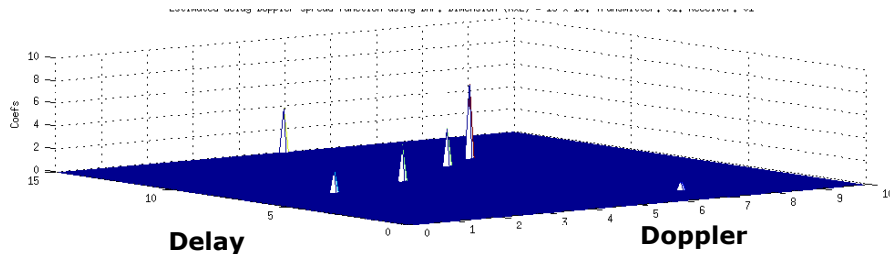


Figure 6–11: 2×2 MIMO case. Delay-Doppler Spread Function $\mathbf{U}_{1,1}$ estimated using Matching Pursuit Algorithms

6.4 Parallel Modeling Tools

6.4.1 pMatlab

The computational complexity signal processing algorithms can be excessive, even more so when they are manipulated signals with a high number of samples (order 2^{20}). Knowing also that many applications, such as radar and sonar applications, require real-time responses, it is mandatory to develop high-performance computational applications to improve the response times. Multicore architectures are now available for application developers, giving the possibility to exploit the power of parallel systems at a reasonable cost. Simultaneously, the industry of the software has developed tools that allow it to make the application conversion serial-to-parallel with low level of difficulty. One of these tools is pMatlab, a parallel toolbox developed for Matlab, in Lincoln Laboratory, of MIT. It allows to make parallel implementations in a fast and efficient way.

`%Create a 4x4 dmat object of ones with map Map`

D(0,0)=1	D(1,0)=1	D(2,0)=1	D(3,0)=1
D(0,1)=1	D(1,1)=1	D(2,1)=1	D(3,1)=1
D(0,2)=1	D(1,2)=1	D(2,2)=1	D(3,2)=1
D(0,3)=1	D(1,3)=1	D(2,3)=1	D(3,3)=1
Processor 0	Processor 1	Processor 2	Processor 3

Figure 6–12: How *dmat* datatype distribute a bi-dimensional array between 4 processors

pMatlab can be considered a framework of parallel programming based on PGAS (Partitioned Global Address Space) [46]. This category exploits the creation mechanisms of global arrays to be distributed in more than one processor. This approach allows the programmer to use the distributed array like it was an unique object, therefore, making very comfortable the migration from serial-to-parallel approach. In the case of the ambiguity function, there is a natural approach to the

PGAS paradigm, making of pMatlab a very attractive option for its implementation. Many efforts have been made to develop fast computation algorithms for ambiguity function.

A special connection with Kronecker products algebra has been sought. However, there are some complications for algorithm parallel implementation. Among them we can point out: keeping track of which processor data is located in, determining which processor is free for working, distributing data such that communication between processors is minimized, distributing computation such that performance per processor is maximized, synchronizing data between processors (if necessary), debugging communication between processors.

pMatlab introduces a new datatype: the distributed matrix *dmat* is the fundamental data storage datatype in pMatlab, equivalent to *double* in matlab. pMatlab supports *dmat* objects with two, three, or four dimensions. *dmat* objects must be explicitly constructed in pMatlab via a constructor functions (overload methods with additional *map* parameter). The *map* parameter accepts a *map* object which describes how to distribute the *dmat* object across multiple processors. Figures 6–12 and 6–13 show how *dmat* datatype distribute arrays with two and tree dimensions. Figure 6–12 is the outcome of performing the pMatlab code shows in Table 6–10, and Figure 6–13 is the outcome of performing the pMatlab code shows in Table 6–11.

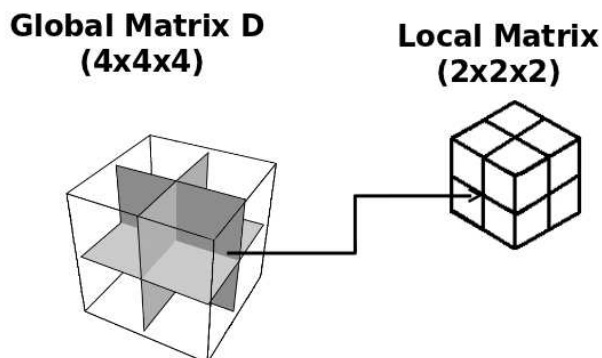
Table 6–10: Script to Create a 2-D Mapping

Line	Instructions
1	<code>Map = map([2 2], , [0:3])</code>
2	<code>D = ones(4, Map);</code>

Table 6–11: Script to Create a 3-D Mapping

Line	Instructions
1	<code>Map = map([2 2 2], , [0:7])</code>
2	<code>D = ones(4,4,4, Map);</code>

```
%Create a 4x4x4 dmat object of zeros using map Map
```

Figure 6–13: How *dmat* datatype distribute a 3-dimensional array between 8 processors

6.5 MIMO Channel Estimation Parallel Approach

The increasing demand for computational power has forced developers to adapt their software application development techniques to the new challenges of increasing speed up demands. Multiple options have arrived to the software market. A big central challenge to developers is represented by the necessary soft-transition from serial to parallel paradigms. This gap has been reduced using tools that, developing intelligent approaches to translate serial-to-parallel code without significant changes in the program and data structures.

In the modeling and estimation fields Matlab has been a powerful tool and a dominant standard. Clear and efficient code, high variety of multi-discipline toolboxes, and a growing number of graphical wizards have positioned Matlab in an important place in sciences and engineering research. Lincoln Lab, in MIT, developed a parallel toolbox for Matlab called pMatlab [47]. High-productivity software

has been developed using pMatlab as parallel platform [48], and many other high-performance DSP applications are based on pMatlab [49]. In high performance computing, Message Pass Interface (MPI) is a very important communication standard for applications working on multi-processor platforms. pMatlab uses a particular MPI distribution called MatlabMPI. Whole version of MatlabMPI was developed in Matlab code.

The idea behind parallel programming is to break down a big problem into smaller problems with a well defined connections map, for monitoring the information flow in a very clear manner. A simple but efficient approach to address this problem consists in dividing the input data in \mathbf{N}_P segments, where \mathbf{N}_P represents the available number of available processors, and then to distribute these segments among processors. This approach is known as *Single Program Multiple Data* (SMPD) paradigm. Each processor P_i runs the same copy of the program; ie, each data segment receives the same treatment in the application level.

In the MIMO ALS estimation problem studied in the previous section, many obstacles for developing a parallel approach are found. First, the process nature is inherently sequential, each stage is strongly related to the previous stages. Second, the amount of data to distribute among processors is significantly abundant. *Matching pursuit algorithm* performs the follow steps in an iterative manner as shows Table 6–12.

Table 6–12: Matching Pursuit Algorithm

Line	Pseudo-Instructions
1	Selects a dictionary column c_i using a maximization or minimization criteria related to a residual vector \mathbf{g}
2	Computes a coefficient λ_i associated with c_i and \mathbf{g}
3	Updates \mathbf{g} vector subtracting the $\lambda_i c_i$ contribution from \mathbf{g} ($\mathbf{g} \leftarrow \mathbf{g} - \lambda_i c_i$)
4	Verifies a stopping condition

In Table 6–12 Steps 1 and 2 perform the more expensive computational effort in the algorithm, so it would be important to apply a possible parallel approach in these specific steps. The inherent sequential nature of step 2 makes it difficult to apply using a parallel approach. However, it is viable to perform a dynamic break down of the recalculating process of λ_i coefficients in the *orthogonal matching pursuit algorithm*. The step 1 is performed through the **Max_Proj** Matlab function (Table 6–13). This function computes the c_i column with maximum contribution (projection) on a **g** residual vector.

Table 6–13: Max_Proj Matlab Function

Line	Instructions
1	<code>function column = Max_Proj(C,g,chosen_columns)</code>
2	<code>columns = size(C,2);</code>
3	<code>j = setdiff([1:columns],chosen_columns);</code>
4	<code>R = C(:,j);</code>
5	<code>proj = ((R' * g).^2) ./ diag(R' * R);</code>
6	<code>[Max,column] = max(proj);</code>
7	<code>end</code>

In the **Max_Proj** Matlab function, the **R** matrix must be distributed in order to apply a parallel computational approach. This procedure will be applied using pMatlab tools. First we need to declare the **R** matrix as *distributed matrix datatype*. For this purpose, it is necessary to perform the pMatlab instruction *map*. Table 6–14 shows the pMatlab *Distributed Map* definition, *distributed matrix object* construction, and *distributed matrix* initialization, based on the sequential code presented in Table 6–13.

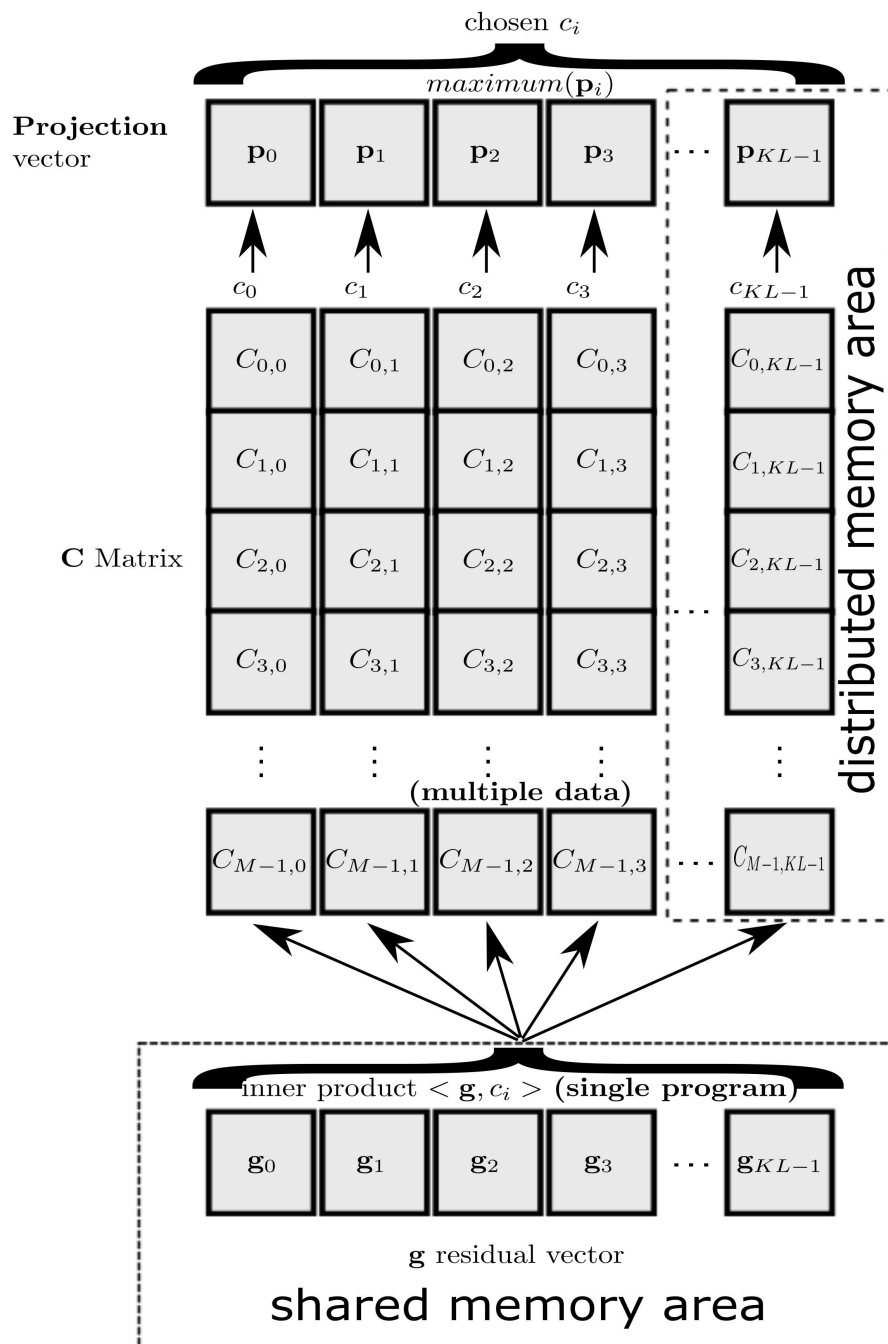


Figure 6–14: Single Program Multiple Data Paradigm Applied to Selection Column Procedure in OMP and BMP Algorithms

Now C matrix data has been loaded in a distributed data structure R and it can be separated and treated for processing in each individual available processor. This approach is based on *Single Program Multiple Data* paradigm. Table 6–15 shows

Table 6–14: pMatlab Distributed Map Definition

Line	Instructions
1	<code>[rows, columns]=size(C);</code>
2	<code>mapR = map[1 Ncpus], {}, [0:Ncpus-1];</code>
3	<code>j = setdiff([1:columns], chosen_columns);</code>
4	<code>R = zeros(rows, length(j), mapR);</code>
5	<code>R = C(:, j);</code>

Table 6–15: Matlab/pMatlab Code for Distributed Column Selection

Line	Instructions
1	<code>mapP = map([Ncpus 1], {}, [0:Ncpus-1]);</code> <i>% mapping design $\rightarrow \mathcal{C}^{V \times P(D \times L)}$</i>
2	<code>proj=zeros(rows, 1, mapP);</code> <i>% proj : $\mathcal{C}^{V \times P(D \times L)}$</i>
3	<code>p = local(proj)</code> <i>% $p \leftarrow proj.loc$</i>
4	<code>r = local(R);</code> <i>% $r \leftarrow R.loc$ (scattering)</i>
5	<code>p = ((r' * g).^2) ./ diag(r' * r);;</code> <i>% local computation</i>
6	<code>proj = put_local(p);</code> <i>% $proj.loc \leftarrow p$</i>
7	<code>proj = agg(proj);</code> <i>% gathering</i>
8	<code>[Max, column] = max(proj);</code> <i>% getting result</i>

the sequence of Matlab/pMatlab instructions necessary to complete the column selection process, in line 1 the mapping is designed, the mathematical nomenclature for this map implementation is defined as $proj : \mathcal{C}^{V \times P(D \times L)}$, where the P operator defines the blocks scattering map on the $D \times L$ columns, and \mathcal{C} defines the complex data type. The whole selection column procedure is illustrated in Figure 6–14. In this illustration, the shared and distributed memory areas can be seen.

This parallel approach can be implemented in any development parallel tool. However, pMatlab offers a simple way to code implementations without incurring

too much code overhead compared to the original sequential code. The *distributed matrix data type* is a powerful tool for fast parallel implementation of proofed sequential algorithms, like the **Max_Proj** function for column selection in *basic matching pursuit* and *orthogonal matching pursuit* algorithms. The timeline in serial and parallel approaches is shown in the Figure 6–15. This approach is not valid for *order recursive least square matching pursuit* since it uses a different criteria for column selection, based on minimization of \mathbf{g} residual norm. When delay-Doppler resolution grows, this parallel approach offers significant improvements in the algorithm speed up process. High delay-Doppler resolution is useful for improving the parameter detection. Therefore, it is necessary to address parallel approaches for ALS channel parameter estimation.

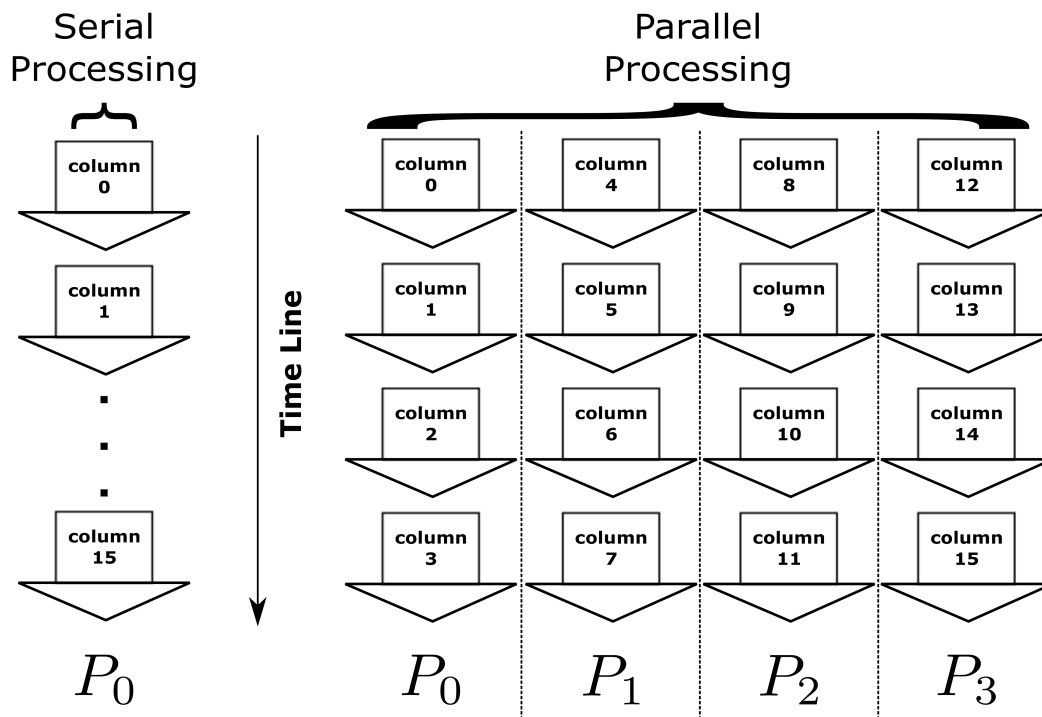


Figure 6–15: Timeline in Serial and Parallel Approaches. Case: 16 Columns and 4 Processors

In this approach, the *delay-Doppler resolution* depends on the D , and L parameters. The computational complexity of **Max_Proj** column selection procedure

is $\mathbf{O}(V \times D \times L)$, this load can be divided among N_{cpus} available processors. This procedure is core in *basic matching pursuit* and *orthogonal matching pursuit* algorithms. They are very important tools in the ALS channel parameter estimation process.

6.6 MIMO ALS Parallel Approach

In the MIMO ALS channel case, the data segmentation is natural because the output signal $\mathbf{w}(t)$, represented by the signals collection $[w_0(t) \ w_1(t) \ \dots \ w_{R-1}(t)]$, where R is the number of receivers, can be distributed using a simple segmentation criteria. Remembering that on the MIMO ALS channels the treatment received by each $w_r(t)$ received signal is the same, ie. the procedure method is the same for each received signal using a different data set (*Single Program Multiple Data paradigm* again). This behavior fits in *SPMD* paradigm, conducting to a parallel model implementation. Figure 6–16 shows a MIMO ALS channel structure, it allows to visualize the parallel nature of the channel.

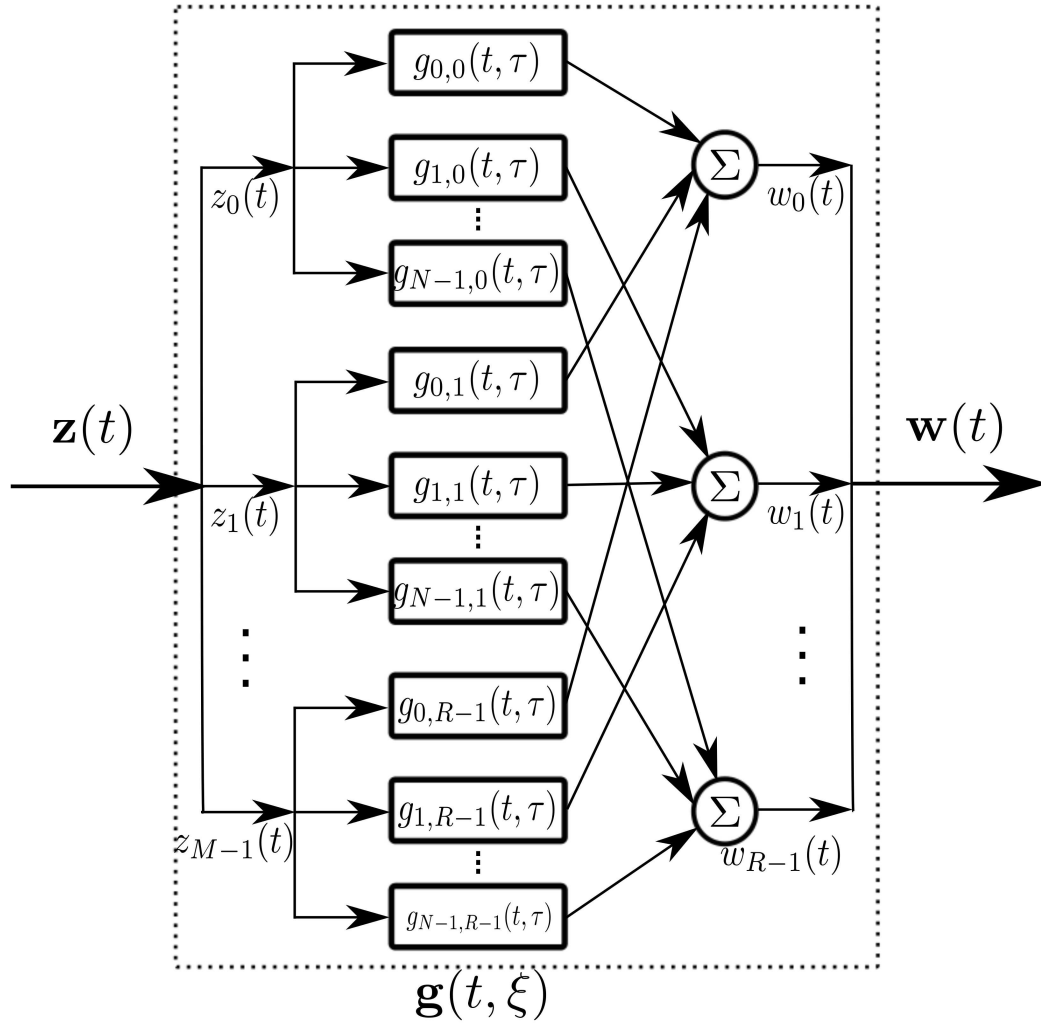


Figure 6–16: MIMO ALS Channel Structure

The MIMO ALS channel estimation problem is reduced to estimate efficiently a SISO ALS channel and develop an accurate procedure for data segmentation, breaking it down in their collection elements $[w_0(t) \ w_1(t) \ \dots \ w_{R-1}(t)]$. Finally, we need to estimate each SISO ALS channel and gather the partial solutions to obtain a global outcome.

6.6.1 Computational Formulation

Each continuous input signal $z_s(t) \in l^2(\mathbb{R})$ can be sampled and represented as a signal $z_s[v] \in l^2(\mathbb{Z}_V)$ and so be described as a discrete signal with the window's length M . Under this approach, we can express the MIMO ALS input-output relationship

as follows:

$$w_r[v] = \sum_{s=0}^{S-1} \sum_{d=0}^{D-1} \sum_{l=0}^{L-1} \alpha_{d,r,s} z_s[v - \xi_{d,r,s}] e^{+j2\pi f_{l,r,s} m}, \quad (6.50)$$

where $d \in \mathbb{Z}_D$ is the number of delays, $l \in \mathbb{Z}_L$ is the number of Dopplers, $r \in \mathbb{Z}_R$ is the number of receivers, $s \in \mathbb{Z}_S$ is the number of transmitters (senders), $m \in \mathbb{Z}_V$ is the length of the window, $w, z \in l^2(\mathbb{Z}_V)$ are the output and input signals respectively.

This representation is computationally feasible because all of its data elements can be represented through computational data structures, and can be loaded in a memory system. The continuous representation can not be represented using discrete structures given its infinite nature. In the next subsection, the data structures related to the MIMO ALS channel problem will be treated.

6.6.2 Kuck's Diagrams Representation

The general formulation of a parallel structure about MIMO ALS channel estimation can be described using Kuck's diagrams. David Kuck presented an abstract manner to represent parallel computational structures. This methodology of representation is nearly related with the *Parallel Random Machine Model* (PRAM) used to describe an abstract parallel machine. a parallel random-access machine (PRAM) is a shared-memory abstract machine. As its name indicates, the PRAM was intended as the parallel-computing analogy to the random-access machine (RAM). Like Random Access Machine (RAM) models, that are used by sequential-algorithm designers to model algorithmic performance, the PRAM is used by parallel-algorithm designers to model parallel algorithmic performance (such as time complexity, where the number of processors is a parameter). PRAM used the concept of shared memory to avoid the communication problem in Message Pass Interface (MPI) models. The complexity is expressed as a function of the dimension of input set N and the number of processors N_{cpus} .

The Kuck's approach creates a hierarchical structure to represent in parallel manner the execution of an algorithm. This hierarchical structure reflects the connection between the computing units and shared memory stages. Each level is tagged with a specific label, like "1.5". This representation approach is sufficiently abstract to allow implementations in a wide spectrum of modern architectures. This fact is very appreciated in a changing world of computational devices. The Figure 6–17 illustrates the Kuck's Diagram Representation of the parallel MIMO ALS estimation problem. In this figure the level 0 presents independent memory units, M_0 , and uniprocessors P_0 , which contain registers and caches for fast access to frequently used data. At level .5 these units are interconnected with a network $N_{.5}$, generally a bus, mesh, hypercube, shuffle, or other mechanism of data interchange. This network provides communication between the processors, but doesn't provide shared memory addressing capabilities. So, a protocol of Message Pass is necessary. The first level of shared-address memory space is SM_1 . The communication between the processors in level 0 and SM_1 is managed by SMN_1 . There exists a great difference between *direct* access to memory via SMN_1 and *indirect* access to memory via $N_{.5}$. The difference is appreciated in a poor performance produced by the *indirect* access to memory. Now, we can apply recursive principle to generate new levels. This recursive approach is employed in the parallel MIMO ALS estimation problem as is described in the Figure 6–17, where each internal diagram represents a SISO problem instance and the whole diagram represents a MIMO problem instance.

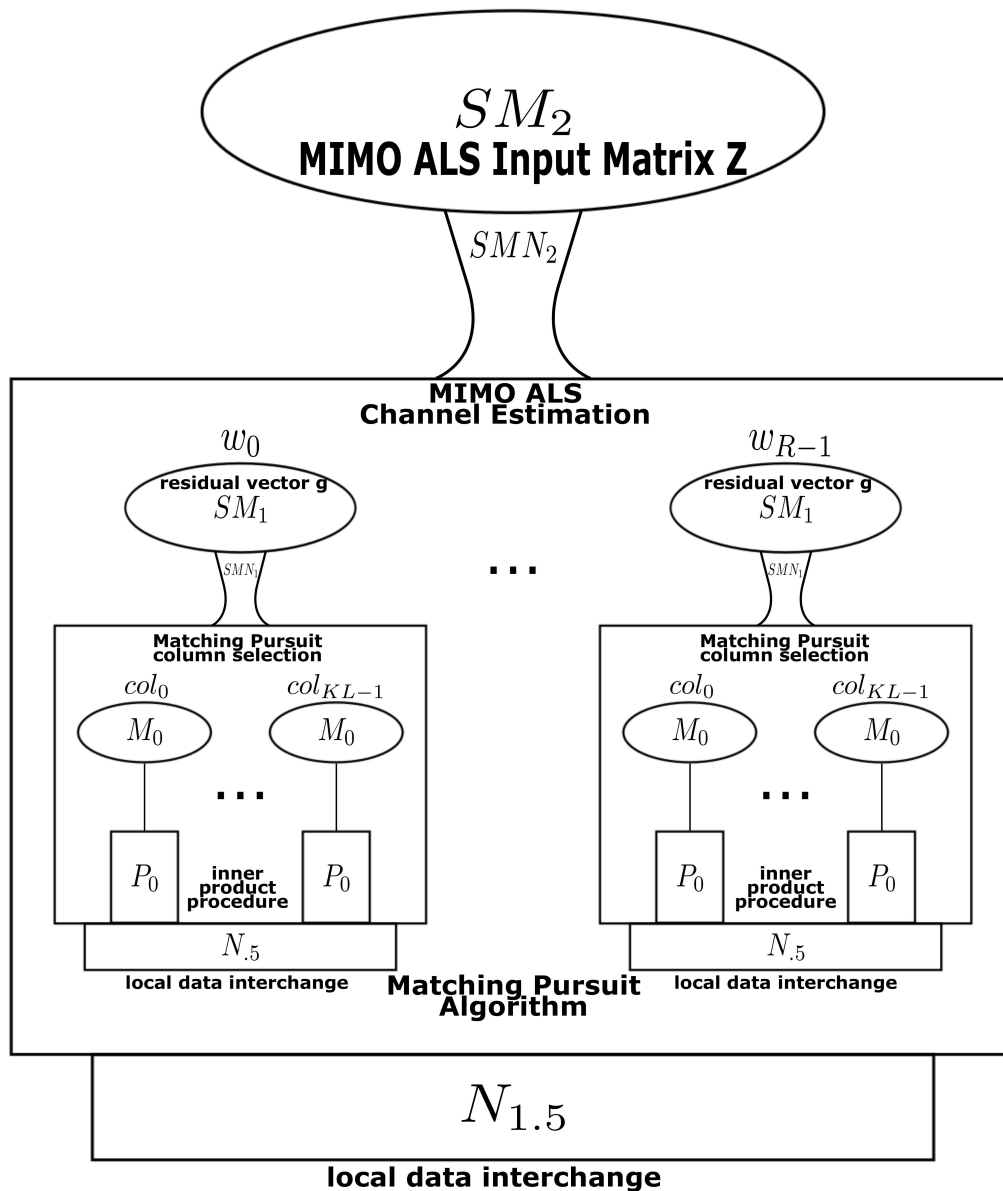


Figure 6–17: Kuck’s Diagram to Parallel MIMO ALS Estimation Problem.

6.6.3 Data Structures

In the previous subsection the discrete formulation of the MIMO ALS channel problem was presented, under this assumption, the input data $\mathbf{z}(t)$ can be represented using a complex bi-dimensional matrix Z of size $\mathbf{V} \times \mathbf{S}$. For this purpose we will create a distributed array Z using pMatlab code showed in Table 6–16, where S is the number of transmit transducers and M is the window’s length of each discrete

input signal $z_s[v] \in l^2(\mathbb{Z}_V)$. The initialization process can be completed doing direct assignment of each input signal $z_s[v]$ on its corresponding column on Z matrix. The Figure 6–18 shows the distribution of input signals $z_v[d]$ inside input matrix Z . We can apply the *scattering data* operation over the matrix Z ; ie, to distribute its columns among available processors. Partial computations will be carried in each processor and finally the gathering process will allow to get an unified solution. This process will be performed in each receiver for calculating the corresponding $w_r(t)$ output signal.

Table 6–16: Script to Create a 2-D Mapping for Distributing the Matrix Z .

Line	Instructions
1	<code>mapZ = map([Ncpus 1], {}, [0:Ncpus-1])</code>
2	<code>% mapping design $\rightarrow \mathcal{C}^{V \times P(S)}$</code>
3	<code>Z = zeros(M,S,mapZ);</code>
4	<code>% Z : $\mathcal{C}^{V \times P(S)}$</code>

$z_0[0]$	$z_1[0]$	$z_2[0]$	$z_3[0]$...	$z_{S-1}[0]$
$z_0[1]$	$z_1[1]$	$z_2[1]$	$z_3[1]$...	$z_{S-1}[1]$
$z_0[2]$	$z_1[2]$	$z_2[2]$	$z_3[2]$...	$z_{S-1}[2]$
$z_0[3]$	$z_1[3]$	$z_2[3]$	$z_3[3]$...	$z_{S-1}[3]$
■	■	■	■		■
■	■	■	■		■
■	■	■	■		■
$z_0[M-1]$	$z_1[M-1]$	$z_2[M-1]$	$z_3[M-1]$...	$z_{S-1}[M-1]$

Figure 6–18: MIMO ALS Input Matrix Z (Size $V \times S$) Containing each Input Signal $z_s[v]$

Another very important structures in this problem are α , ξ , and f matrices. These parameters must be represented using 3-dimensional matrices because their data values depend of the number of transmitter transducers S , the number of receiver transducers R , and the number of delay D or the number of Doppler L . Under this perspective, we need to declare three 3-dimensional distributed matrices **Attenuation**, **Delay**, and **Doppler**. Analyzing the Equation 6.50, we can decouple the former three matrices; ie, we can scatter the data among N_{cpus} available processors sending a complete copy of the Z matrix to each processor. In this scenario each output signal $w_r[v]$ will be a function of the MIMO ALS Z matrix and the correspondent data segment to the output r . The Equation 6.51 shows the functional

relation between w_r , Z , and $Attenuation.loc$, $Delay.loc$, $Doppler.loc$

$$w_r = \mathcal{T}\{Z, Attenuation.loc_r, Delay.loc_r, Doppler.loc_r\}, \quad (6.51)$$

where \mathcal{T} is the transformation to map the input signal Z and parameters (α, ξ, ν) to output signal w , and $Atenuation.loc_r$, $Delay.loc_r$ and $Doppler.loc_r$ are the local parameter segments of α, ξ , and ν parameters for output r . The pMatlab code to parameters data distribution is shown in Table 6–17, where $P(R)$ refers to data partition on the R dimension and \mathbb{R} is the real numbers set. In this point is remarkable that still parameter D can not be equal to parameter L the map formulation is the same in both cases.

Table 6–17: Script to Create a 2-D Mapping for Distributing Channel Parameters.

Line	Instructions
1	<code>map3D = map([1 N 1] , , [0:Ncpus-1]);</code>
2	<code>% mapping design $\rightarrow \mathbb{R}^{D L \times P(R) \times S}$</code>
3	<code>Attenuation = zeros(D,R,S,map3D);</code>
4	<code>% Attenuation : $\mathbb{R}^{D \times P(R) \times S}$</code>
5	<code>Delay = zeros(D,R,S,map3D);</code>
6	<code>% Delay : $\mathbb{R}^{D \times P(R) \times S}$</code>
7	<code>Doppler = zeros(L,R,S,map3D);</code>
8	<code>% Doppler : $\mathbb{R}^{L \times P(R) \times S}$</code>

The more important action in the previous pMatlab declaration is related to data distribution. We are assuming the existence of N process units available (processors) and we are distributing each of matrices **Attenuation**, **Delay**, and **Doppler** among the $Ncpus$ processors, ie., each matrix slide will be treated in its correspondent processor $ncpu$. The mapping is defined in the first instruction, where the `[1 N 1]` parameter indicates that the matrix will be distributed dividing the

second dimension (R) including all elements of each slide (partial matrices of size $D \times S$ or $L \times S$). Mapping definition is the initial step for a parallel approach pMatlab-based.

The next step is associated with data scattering. Each process must receive a copy of the data segment to be processed locally. Using the 'local' instruction each process can obtain its data segment. The syntax is presented in Table 6–18, where the segment of *dmat* correspondent to the processor *ncpu* was copied to the variable `local`. Therefore, it can be treated as a regular variable (non-distributed). The Figure 6–19 shows the data partition applying the map *map3D* on the 3D attenuation matrix. At this point the data scattering has been completed. It is possible to obtain local indexes for the local data segment using the `get_local_ind` command. Now we are ready for processing the input signal $z_r[v]$ in the local processor *ncpu*.

Table 6–18: Initialization of Local Variables.

Line	Syntax
1	<code>attenuation_loc = local(Attenuation);</code>
2	<code>% Attenuation.loc ← local(Attenuation)</code>
3	<code>delay_loc = local(Delay);</code>
4	<code>% Delay.loc ← local(Delay)</code>
5	<code>doppler_loc = local(Doppler);</code>
6	<code>% Doppler.loc ← local(Doppler)</code>

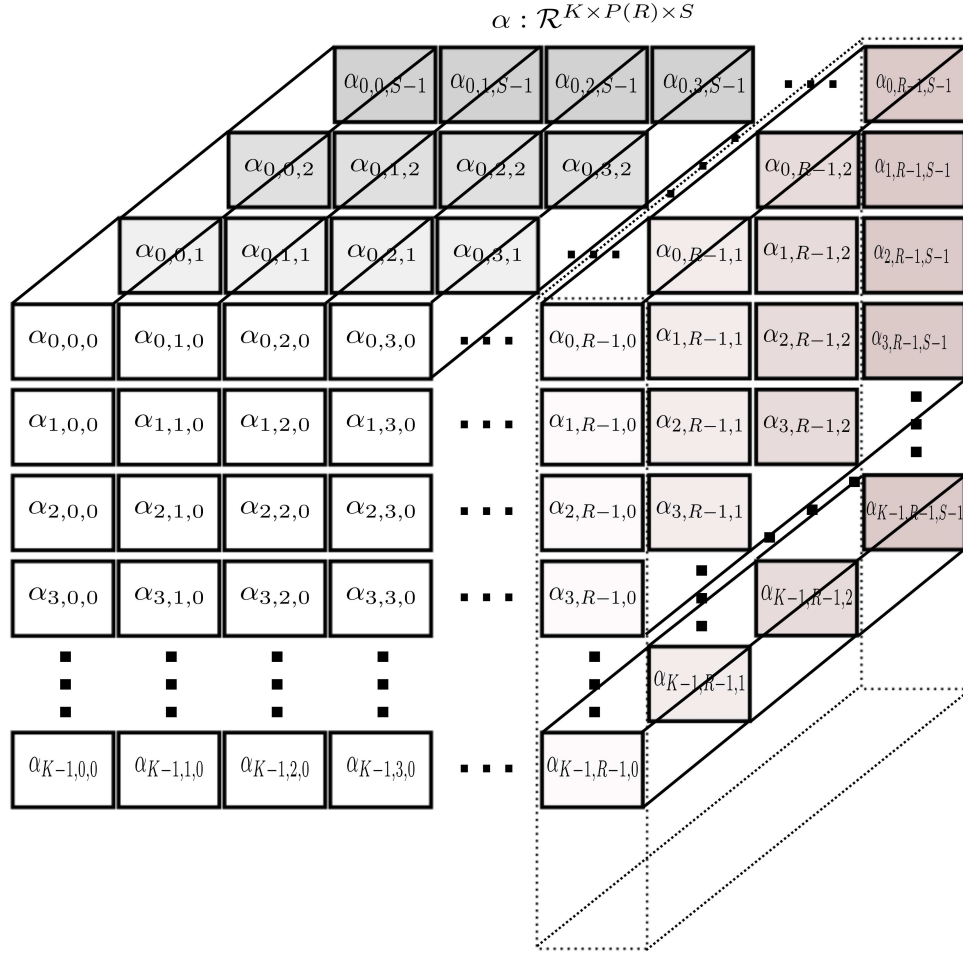


Figure 6–19: Data partition on 3D Attenuation Matrix

6.7 Testbeds

Modeling underwater acoustic media for signal analysis is a very complex process. For this reason, experimental platforms in the form of testbeds are developed to assist in difficult tasks, such as proof of concept demonstrations, theory testing, principle validation, new concept generation, and computational tool enhancement. We were able to establish an experimental platform as a computational testbed to test our formulated subsystems conforming our proposed CSP modeling framework. The experimental platform consisted of two 48GB workstations interconnected via a network switch and dedicated gigabit Ethernet links.

Figure 6–24 depicts the experimental platform configured to platform testbed analysis of MIMO ALS channel subsystems using multidimensional, multicomponent polynomial phase signals, as well as the use of pseudo-random Gaussian noise (PRGN) waveforms, as sounding signals to estimate the delay-Doppler spread function $U(\xi, \nu)$.

The experimental platform was also configured to conduct testbed analysis for the ISS operator sub-systems. Figure 6–25 depicts a computer-based ISS testbed used for multi-target search, detection, estimation, and tracking operations. Figures 6–20 and 6–21 show 3D and 2D representation, respectively, of the response of an ISS subsystem when a search pulse, with length 16,536 complex samples, is transmitted and the scattered echo signal is received. Figure 6–22 and Figure 6–23 represent the same process when the transmitted signal is substituted for a linear chirp signal.

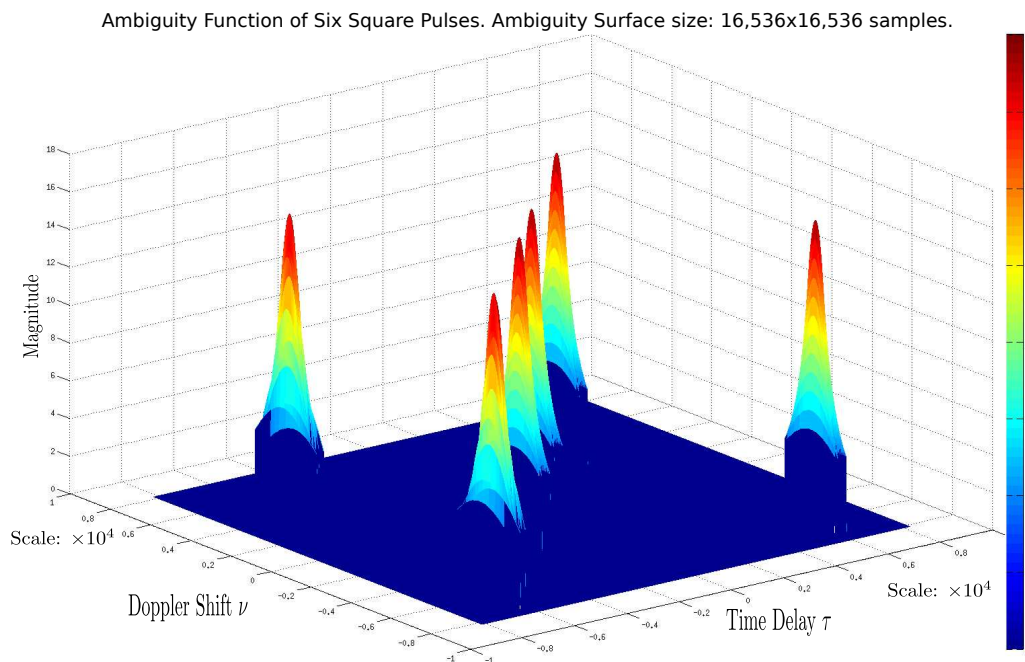


Figure 6–20: Ambiguity Function Surface $A_z \in l^2(\mathbb{Z}_{2^{16}})$ in a 3D Representation. Six Square Pulses.

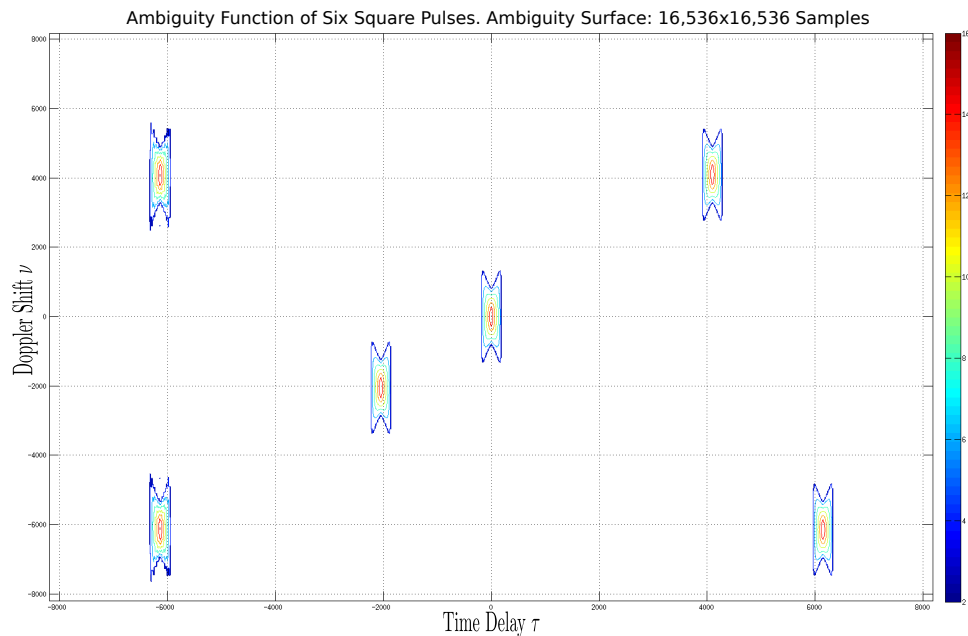


Figure 6–21: Ambiguity Function Surface $A_z \in l^2(\mathbb{Z}_{2^{16}})$ in a 2D Representation. Six Square Pulses.

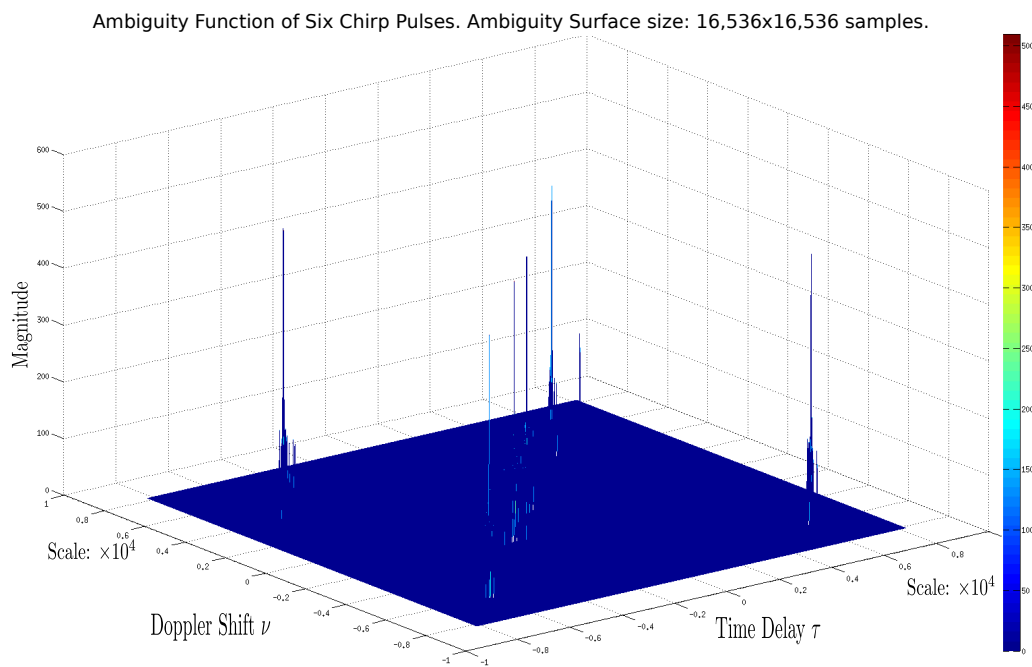


Figure 6–22: Ambiguity Function Surface $A_z \in l^2(\mathbb{Z}_{2^{16}})$. Chirp Pulse.

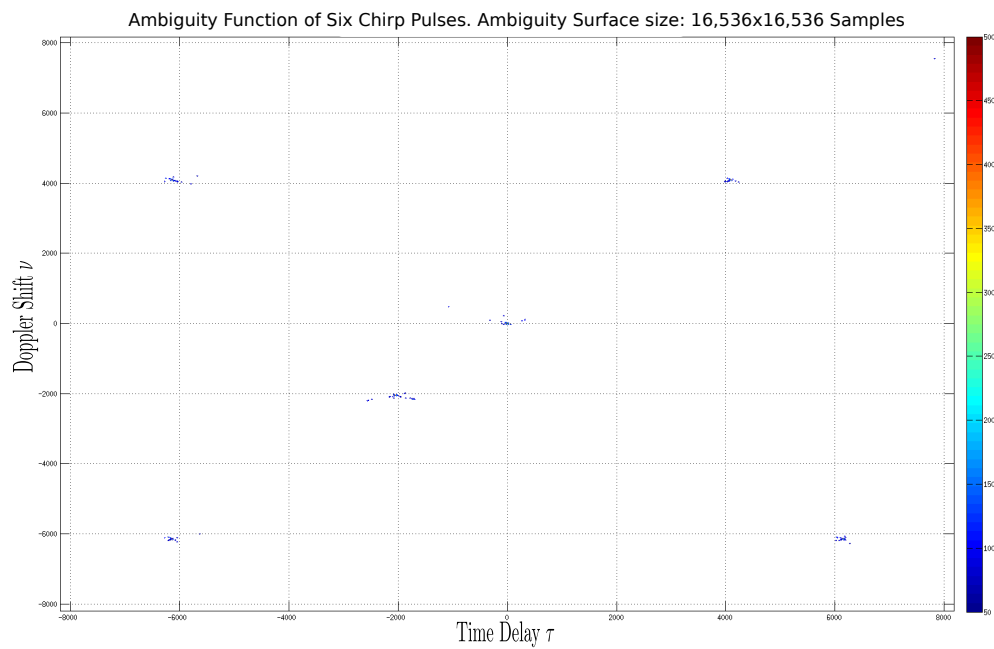
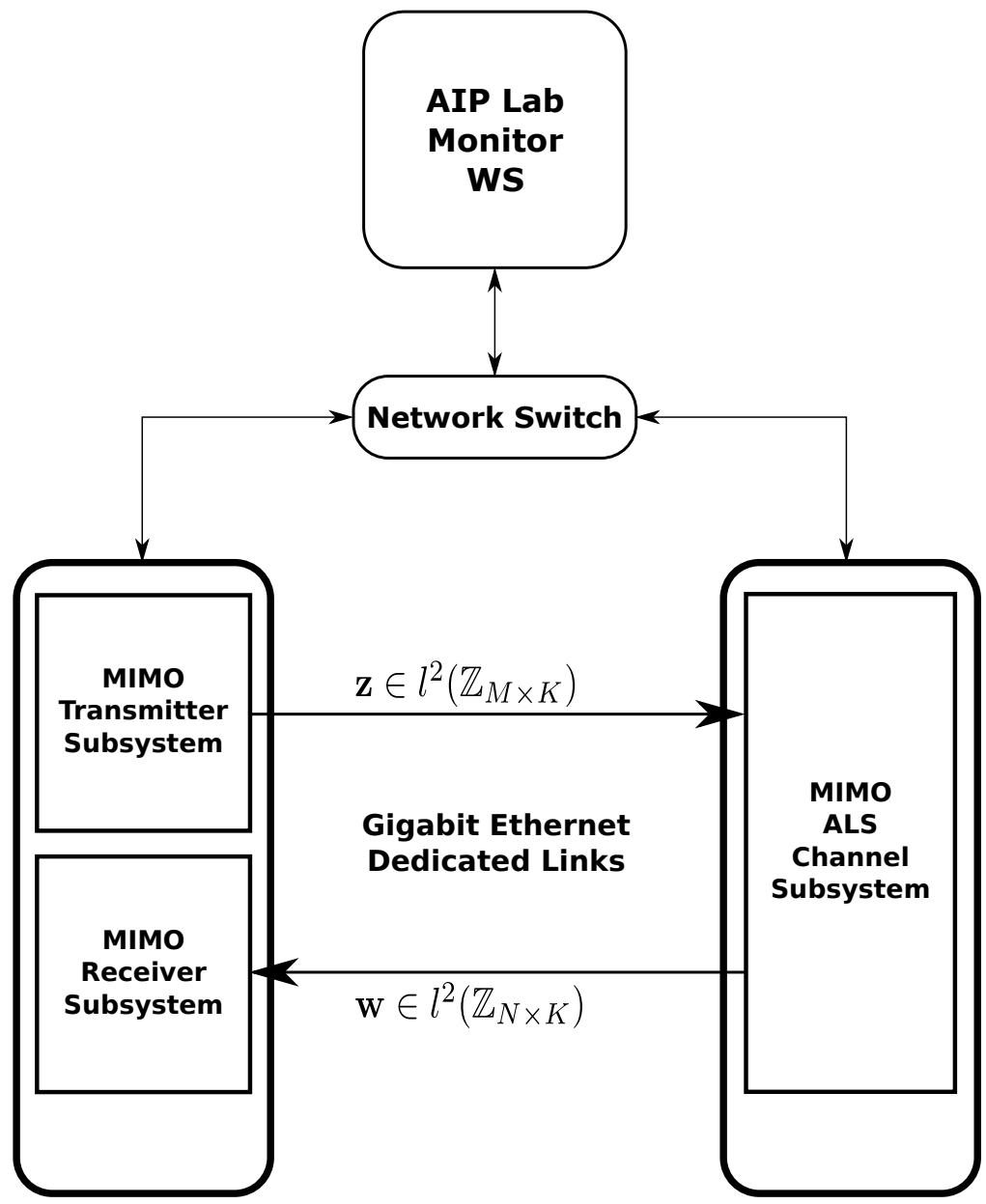


Figure 6–23: Ambiguity Function Surface $A_z \in l^2(\mathbb{Z}_{2^{16}})$ in a 2D Representation. Chirp Pulse.



WS-1
 Dual-Quad Core @ 3.2GHz - Dell Precision Workstations (WS)
 Model: T7500 - 48GB SDRAM @ 1066 MHz - 3Gbps SAS NIC

WS-2

Figure 6-24: Computer-Based ALS MIMO Channel Testbed. WS-1 and WS-2 are Dell Precision T7500 Workstations with Dual-Quad Core and 48GB RAM

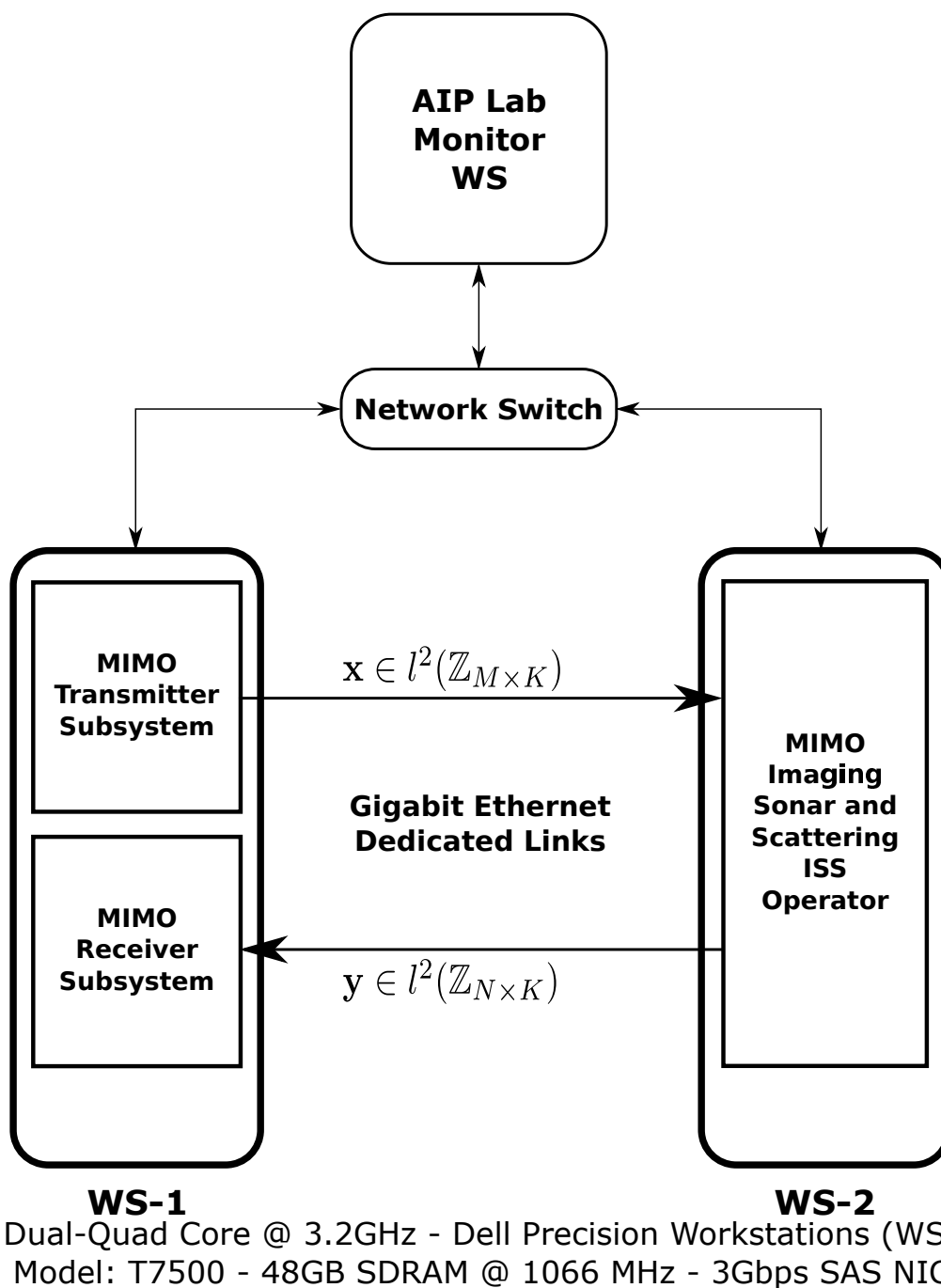


Figure 6–25: Computer-Based ISS Testbed. WS-1 and WS-2 are Dell Precision T7500 Workstations with Dual-Quad Core and 48GB RAM

ALSISS System

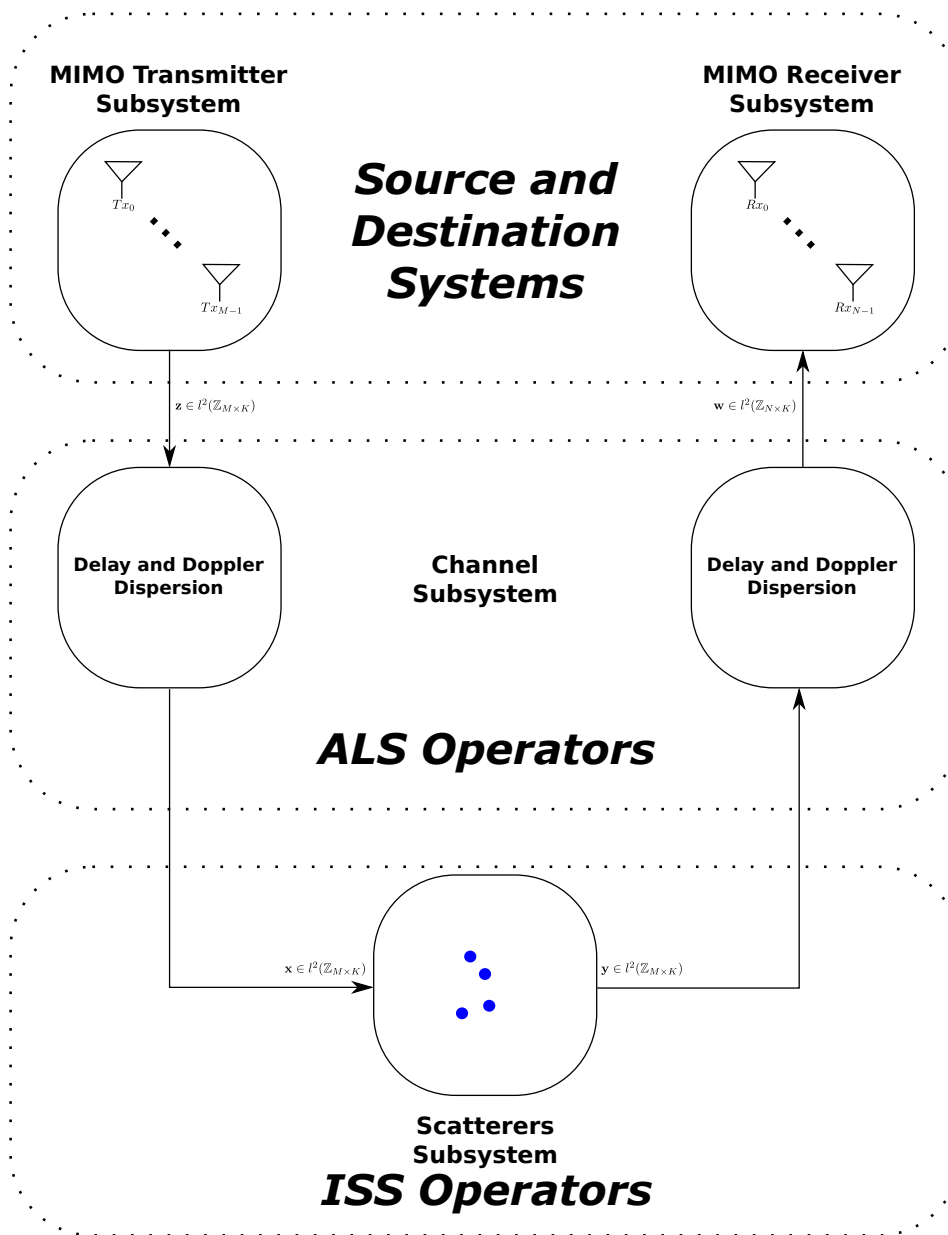


Figure 6–26: Integrated ALS-ISS System

7. Ethical Considerations

7.1 Introduction

This chapter deals with the ethical considerations that every doctoral student must address during his/her doctoral work and after completion of such work, when he/she begins his/her professional research work as part of society. Ethics defines the set of rules and moral principles that govern a person's behavior in society and determines the moral correctness of a specified conduct. We will address in our work these ethical considerations as well as ramifications involving environmental and legal issues. We will also address other potential considerations that may affect the ethical conduct of our work.

7.2 Sea and Ecosystems

From early ages the study of the sea and its ecosystems has been a topic that has inspired to the human beings, both from a scientific as well as poetic dimension. The vastness of the oceans has boosted the development of myths and legends that have slowly been unveiled with the passage of time and the advancement of the science. The knowledge of the sea has changed the way of thinking in our modern world, however, the man in an effort to explore more of the sea has encountered some problems. Perhaps the main problem was how to make a complete study of the sea without disturb the underwater ecosystems. Many animals and plants are severely affected with experiments conducted in certain marine ecosystems. A typical example was the nuclear tests conducted by France at Mururoa Atoll. From the experience gained in the conquest of the natural world in the mainland, the man acknowledged that him must be careful in trying to conquer the marine world. We must remember that two thirds of the planet is covered with water. Under this perspective we want to contextualize the development of our thesis work.

Lake Maracaibo is the biggest lake in South-America. In the periphery of it many industries and factories have been installed during the last 80 years. All residuals of industrial processes are thought to the lake. This situation has produce an elevated level of pollution in it. The oil exploration has required to use many time-frequency techniques for characterizing the subsoil under lake. These tests have affected the livings in the lake. I hope to make a small contribution to avoid this problem. The science and the ecology must work together.

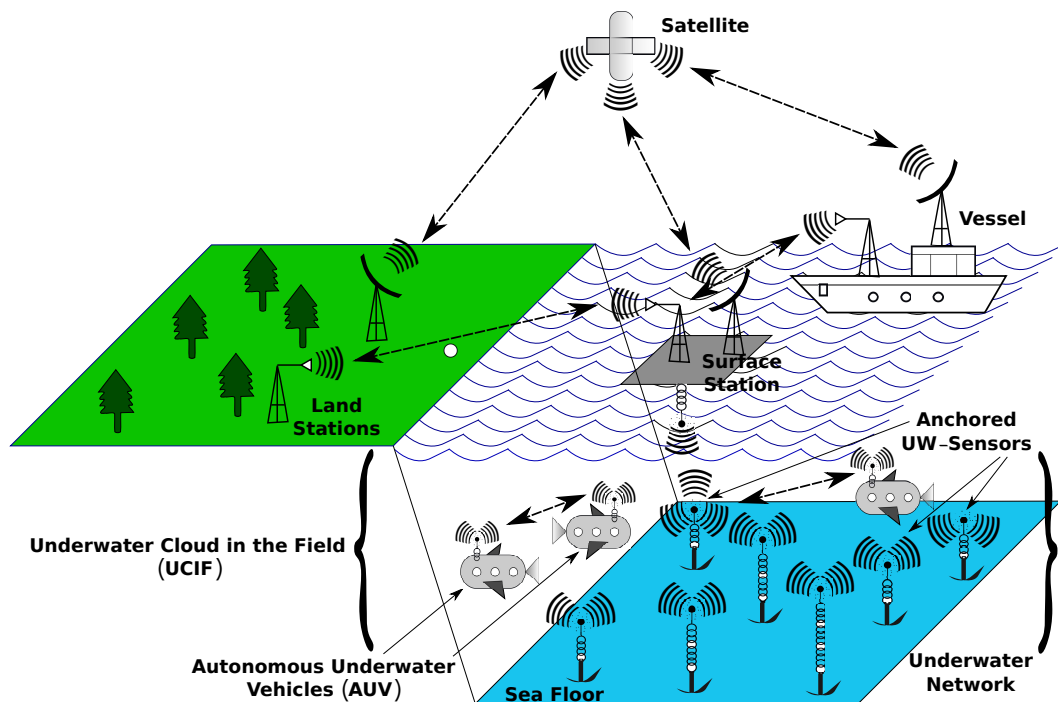


Figure 7-1: Underwater Ecosystem Impacted by ALS Channels Research

7.3 Environment Issues

Many research is performing in the area of environmental impact of the acoustic signals in different groups of marine communities, like whales and dolphins [50][51][52].

Our work deals with the transmission and reception of acoustical information. This mode of transmission affects all living organisms since it deals with the transfer of energy through longitudinal waves pressure waves. Since bio-acoustics is the

study of sound produced by or affecting living organisms, in particular, sounds pertaining to communications, it is of paramount importance to study how any proposed acoustical communication technique may affect the living. Figure 7-1 illustrates the risk associated with acoustic pollution in underwater ecosystems. It can be induced by communication equipment used in underwater communication.

Environmental ethics is the part of environmental philosophy which considers extending the traditional boundaries of ethics from solely including humans to including the non-human world. This research work involves ethical issues to be addressed when the ecosystem is when the ecosystem is significantly affected by the use of acoustic signals that disrupt the lifestyle of living organisms within the habitat. Some disciplines related with this perspective are environmental law, environmental sociology, ecotheology, ecological economics, ecology and environmental geography.

In 2003, an ecologist team called Greenpeace began a campaign to address low-frequency active sonar's effect on marine life. For a number of years, Greenpeace had raised concerns that various types of active sonar were increasing noise levels in the world's oceans to the point that caused physical damage to marine life (www.greenpeacefoundation.org). Before Greenpeace's announcement, the United States military began use of low-frequency, long range active sonar in a special program to acoustically *light* oceans for advance warning of submarine and ship activity by other countries. In 2003 Greenpeace made a new suggestion, recommending acoustic daylight imaging as a more environmentally friendly type of sonar.

Michael Jasny establishes the problem in the following terms: *Undersea noise pollution is like the death of a thousand cuts* [53]. Many whales and other aquatic species depend on sound as they hunt for food, detect predators, find mates, and maintain their awareness in the darkness of the sea. The use of the acoustic signals, that cause noise in the natural communication processes, can produce chaos in underwater communities, like dolphins or whales. On land environments animals

alike know to move away from a loud or traumatic sound; the further we get, the more the sound dissipates. In underwater environments is so different. Sonar and ship noise can send a pulse wave of noise for great distances. It is difficult to pinpoint the origin or source of a particular sound and even harder to avoid it. Whales, dolphins and other marine mammals that have been caught in the wake of sonar have been disoriented during hours. These observations were done by the Oceanic Preservation Society in its official website www.opsociety.org.

7.4 Legal Issues

Legal issue become very relevant when dealing with research involving human subjects. Even though our proposed research does not involve human beings as participants in any systematic experimentation, we are very much aware of the cautionary actions and steps that must be taken in order to respect and preserve the dignity, bodily integrity, autonomy, and privacy of humans.

7.5 Other Considerations

The development of modern societies depend to a large extent upon the contribution of technology, such as the development and use of monitoring and communication systems. These processes are, in general, associated with the production of waves, some of which are unavoidable hazardous. Such radiation requires careful management to ensure adequate protection of living and the environment. We will be very observant during our proposed research work of any other considerations which may surface and may have a potential ethical or legal impact.

8. Conclusions and Future Works

8.1 Conclusions

This thesis presented a *Computational Signal Processing* modeling framework, named ALSISS, for the analysis of underwater acoustic signals associated with operations pertaining to the *search, detection, estimation, and tracking* of underwater moving objects. The thesis concentrated on the use of concepts, tools, methods, and rules from the field of *Information-Based Complexity* to formulate error-approximation algorithms, under a general framework, in order to address underwater acoustics signals analysis problems that could be solved approximately. *Time-frequency calculus* was utilized as the main computational signal processing tool to treat signal analysis problems under the formulated ALSISS framework. Concrete algorithm implementation results were realized using *convex optimization* techniques. New variants of multidimensional *matching pursuit* algorithms for time-variant double dispersive acoustic channel characterization were formulated under the language of *Kronecker Products Algebra* and implemented using the *pMatlab* parallel programming environment. An improved performance of 4-times was obtained with respect to existing algorithms. Some original contributions are presented in the list bellow:

- i Extension Spaces Formulation of J. F. Traub IBC Framework
- ii Parallel Variant of Matching Pursuit Algorithms
- iii Parallel Kronecker Representation of the Ambiguity Function
- iv Integration of ALS and ISS Systems using Operators Theory
- v Unified Representation of Delay-Doppler Spread Function
- vi New Scattering Function Estimation Techniques
- vii Application of Error Approximation Algorithms to Channel Estimation Problems
- viii Development of a New Computational Framework

Most of presented graphs and tables are original and have been created for this thesis work. Some graphs and tables have been reproduced from other works. These reproduced graphs and tables were used to describe previous works.

8.2 Future Works

As future works we present the following considerations:

- i Attempt to fully integrate main concepts tools, methods, and rules of *Information-Based Complexity* (IBC) into the time-frequency calculus.
- ii Extend the techniques of *Tensor Signal Analysis* and *Kronecker Products Algebra* to better exploit the inherent parallel computational structure of *Multiple-Input Multiple-Output* (MIMO) matching pursuit algorithms.
- iii Seek a more systematic and automated integration environment between the ALS-ISS modeling framework and the associated testbeds described in the thesis.
- iv Extend the work on *Computable Analysis* in order to better characterize *real-number models of computation* suitable for time-frequency calculus.
- v Extend the mathematical analysis of waveform design techniques by using Weyl-Heisenberg group theory.
- vi Improve on channel estimation techniques for double dispersive channels by using advanced time-frequency calculus.
- vii Extend the ALS-ISS system integration framework by incorporating non-additive measures.
- viii Improve on optimal error algorithm analysis by using Traub's adversary principle.

Bibliography

This bibliography includes representative set of textbooks consulted as part of the literature review for this research.

1. Blahut, Richard E., et al., *Radar and Sonar. Part I*, Springer-Verlag, New York, ISBN: 0-387-97516-0.
2. Blum, Lenore, et al. *Complexity and Real Computation*, Springer-Verlag, New York, ISBN: 0-387-98281-7.
3. Boyd, Stephen, and Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, ISBN: 978-0-521-83378-3.
4. Christakos, George, *Random Field Models in Earth Sciences*, Dover Publications Inc., New York, ISBN: 0-486-43872-4.
5. Hancock, John C., and Wintz, Paul A., *Signal Detection Theory*, McGraw-Hill Inc., New York, Library of Congress Catalog Card Number: 66-19462-25982.
6. Hopcroft, John E., et al., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, USA, ISBN-10: 0321455363.
7. Jeffress, Lloyd A., *The Hixon Symposium: Cerebral Mechanism in Behavior*, John Wiley & Sons Inc., New York.
8. Kepner, Jeremy, *Parallel Matlab for Multicore and Multinode Computers*, SIAM, Philadelphia, ISBN: 978-0-898716-73-3.
9. Kuck, David J., *High Performance Computing. Challenges for Future Systems*, Oxford University Press, New York, ISBN: 0-19-509551-0.

10. Levanon, Nadav, and Mozeson Eli, *Radar Signals*, Willey & Sons Inc., New Jersey, ISBN: 0-471-47378-2.
11. Li, Ming, and Vitányi, Paul, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer-Verlag, New York, ISBN: 0-387-94868-6.
12. Mahafza, Bassem R., *Radar Systems Analysis and Design Using MATLAB*, Chapman & Hall/CRC, Boca Raton, ISBN-10: 1-58488-532-7.
13. Khinchin, A. I., *Mathematical Foundations of Information Theory*, Dover Publications Inc., New York, Library of Congress Catalog Number: 57-13025.
14. Peebles, Peyton Z., *Probability, Random Variables, and Random Signal Principles*, McGraw-Hill Inc., Princeton, ISBN: 0-07-049273-5.
15. Pillai, S. Unnikrishna, *Array Signal Processing*, Springer-Verlag, New York, ISBN: 0-387-96951-9.
16. Plaskota, Leszek, *Noisy Information and Computational Complexity*, Cambridge University Press, Cambridge, ISBN: 0-521-553368-7.
17. Sidiropoulos, N. D., and Gershman, A. B., *Space-Time Processing for MIMO Communications*, Wiley & Sons Inc., Southern Gate, England, ISBN-10: 0-470-01002-9.
18. Sontag, E. D., *Mathematical Control Theory. Deterministic Finite Dimensional Systems*, Springer-Verlag, New York, ISBN: 0-387-97366-4.
19. Tolimieri, Richard, et al., *Algorithms for Discrete Fourier Transform and Convolution*, Springer-Verlag, New York, ISBN: 0-387-98261-2.
20. Tolimieri, Richard, and An, Myoung, *Time-Frequency Representations*, Birkhäuser, Berlin, Germany, ISBN: 0-8176-3918-7.
21. Traub, J. F., and Werschulz, A. G., *Complexity of Information*, Cambridge University Press, Cambridge, ISBN: 0-521-48005-1.
22. Van Leeuwen, J., *Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity*, MIT Press, Cambridge, ISBN: 0-444-88071-2.

23. Van Loan, Charles, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, ISBN: 978-0-898712-85-8.
24. Von Neumann, John, *The Computer and the Brain*, Yale University Press, London, ISBN: 0-300-02415-0.
25. Wilf, Herbert S., *Algorithms and Complexity*, Prentice-Hall, Inc., New Jersey, ISBN: 0-13-021973-8 025.
26. Wilkinson, Barry, and Allen, Michael, *Parallel Programming. Techniques and Applications Using Networked Workstations and Parallel Computers*, Prentice Hall, New Jersey, ISBN: 0-13-671710-1.

References

- [1] J. F. Traub and Henryk Woźniakowski. Perspectives on information-based complexity. 1992.
- [2] Arthur G Werschulz. An information-based approach to iii-posed problems. *Journal of Complexity*, 3(3):270 – 301, 1987.
- [3] Arthur G Werschulz. An overview of information-based complexity. *Technical Report CUCS-022-02*, 1(1):1–8, 2002.
- [4] Juan Pablo Soto Quiros and Domingo Rodriguez. A computational signal algebra framework for a general class of discrete cohen distributions. *46th Annual Conference on Information Sciences and Systems, CISS 2012*, March 2012.
- [5] F. Hlawatsch and G.F. Boudreaux-Bartels. Linear and quadratic time-frequency signal representations. *Signal Processing Magazine, IEEE*, 9(2):21 –67, April 1992.
- [6] Louis Auslander and R. Tolimieri. *On finite Gabor expansion of signals*, pages 13–23. Springer-Verlag, London, UK, 1990.
- [7] Leon Cohen. Generalized Phase-Space Distribution Functions. *Journal of Mathematical Physics*, 7(5):781–786, 1966.
- [8] Leon Cohen. *Time-Frequency Analysis: Theory and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.
- [9] M.S. Richman, T.W. Parks, and R.G. Shenoy. Discrete-time, discrete-frequency, time-frequency analysis. *Signal Processing, IEEE Transactions on*, 46(6):1517 –1527, June 1998.

- [10] Juan Pablo Soto Quiros. A computational signal algebra framework for a general class of discrete cohen distributions. Master's thesis, University of Puerto Rico, Mayagüez Campus, December 2011.
- [11] J.J. Benedetto and J.J. Donatelli. Ambiguity function and frame-theoretic properties of periodic zero-autocorrelation waveforms. *Selected Topics in Signal Processing, IEEE Journal of*, 1(1):6–20, 2007.
- [12] Gene Bellinger. System: A journey along the way. *A journey in the realm of systems*, 1(1):10–11, 2004.
- [13] George Christakos. *Random Field Models In Earth Sciences*. Dover Publications, Inc., Mineola, NY, USA, 1992.
- [14] C.A. Desoer. Modes in linear circuits. *IRE Transactions on Circuit Theory*, 1(1):1–13, 1959.
- [15] P. Bello. Characterization of randomly time-variant linear channels. *Communications Systems, IEEE Transactions on*, 11(4):360–393, december 1963.
- [16] E.W. Packel and J.F. Traub. Information-based complexity. *Nature*, 327(6125):29–33, July 1987.
- [17] Ray Maleh. Improved rip analysis of orthogonal matching pursuit. *CoRR*, abs/1102.4311, 2011.
- [18] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28:253–263, 2008. 10.1007/s00365-007-9003-x.
- [19] A. Maheshwari and M.A. Dorairangaswamy. Implementation of context free languages in universal turing machines. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 5, pages 332–335, april 2011.
- [20] E.D. Sontag. *Mathematical Control Theory. Deterministic Finite Dimensional Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1990.

- [21] Pan Yingjun and Zou Tie. On the complexity of turing machine accepting fuzzy language. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 112 –116, may 2012.
- [22] Andrew Chi-Chih Yao. Classical physics and the Church–Turing thesis. *J. ACM*, 50(1):100–105, January 2003.
- [23] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.
- [24] H Wotniakowski and William Kaufman. Information-based complexity. *Annual Review of Computer Science, Inc., Palo Alto*, pages 319 – 380, 1986.
- [25] F.J. Traub and A.G. Wershulz. Complexity and information. *Bulletin of the American Mathematical Society*, 37:199 – 204, December 1999.
- [26] Joseph F. Traub and Arthur G. Werschulz. *Complexity and information*. Lezioni Lincee. Cambridge University Press, 1998.
- [27] K. Sikorski. *Optimal Solution of Nonlinear Equations*. Oxford University Press, Jun, 1998.
- [28] A. Keller. *Quasi-Monte Carlo Methods for Photorealistic Image Synthesis*. Shaker, Aachen Press, Apr, 1998.
- [29] K. Frank. *Optimal numerical solution of multivariate integral equations*. Shaker, Aachen Press, Jun, 1997.
- [30] L. Plaskota. *Noisy Information and Computational Complexity*. Cambridge University Press, Jun, 1996.
- [31] A. G. Werschulz. *The Computational Complexity of Differential and Integral Equations: An Information-Based Approach*. Oxford University Press,, May, 1991.
- [32] E. Novak. *Deterministic and Stochastic Error Bound in Numerical Analysis*. Lecture Notes in Mathematics, vol. 1349, Springer-Verlag,, Aug, 1988.

- [33] D. Rodriguez. *Fundamentals of Computational Signal Processing. An Information-Based Complexity Approach*,. Longmans, Green & Co., Puerto Rico, USA, draft edition, July 2012.
- [34] G. Matz and F. Hlawatsch. Time-varying communication channels: Fundamentals recent developments and open problems. In *Proceedings of the 14th European Signal Processing Conference*, Florenz, Italien, September 2006.
- [35] Trym H. Eggen. *Underwater Acoustic Communication Over Doppler Spread Channels*. Massachusetts Institute of Technology Woods Hole Oceanographic Institution. PhD Thesis. Joint Program in Oceanography/Applied Ocean Science and Engineering, Jun, 1997.
- [36] Aydin Akan and Luis F. Chaparro. Modeling and estimation of wireless OFDM channels by using time-frequency analysis. *Circuits Systems and Signal Processing*, 25(3):389–403, 2006.
- [37] G. Matz. Characterization and analysis of doubly dispersive mimo channels. In *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, pages 946 –950, 29 2006-nov. 1 2006.
- [38] A.A.M. Saleh and R. Valenzuela. A statistical model for indoor multipath propagation. *Selected Areas in Communications, IEEE Journal on*, 5(2):128–137, february 1987.
- [39] M. Stojanovic. Adaptive channel estimation for underwater acoustic mimo ofdm systems. In *Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, 2009. DSP/SPE 2009. IEEE 13th*, pages 132 –137, jan. 2009.
- [40] P. Bello. Time-frequency duality. *Information Theory, IEEE Transactions on*, 10(1):18 – 33, jan 1964.
- [41] P.-P.J. Beaujean and L.R. LeBlanc. Adaptive array processing for high-speed acoustic communication in shallow water. *Oceanic Engineering, IEEE Journal*

- of, 29(3):807 – 823, july 2004.
- [42] S.G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397 –3415, dec 1993.
 - [43] Weichang Li and J.C. Preisig. Estimation of rapidly time-varying sparse channels. *Oceanic Engineering, IEEE Journal of*, 32(4):927 –939, oct. 2007.
 - [44] G. Rath and C. Guillemot. Sparse approximation with an orthogonal complementary matching pursuit algorithm. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3325 –3328, april 2009.
 - [45] Jing Xu, Benyong Liu, and Xiang Liao. Speech signal representation via dictionary learning in stft transformed domain. In *Multimedia and Signal Processing (CMSP), 2011 International Conference on*, volume 1, pages 20 –24, may 2011.
 - [46] Bliss N., Kepner J., Kim H., and Reuther A. pmatlab: Parallel matlab library for signal processing applications. *IEEE International Conference*, IV:1189 – 1192, April 2007.
 - [47] N Bliss, J Kepner, H Kim, and A Reuther. pmatlab: Parallel matlab library for signal processing applications. *Acoustics Speech and Signal Processing 2007 ICASSP 2007 IEEE International Conference on*, 4:IV–1189 – IV–1192, 2007.
 - [48] J. Mullen, N. Bliss, R. Bond, J. Kepner, H. Kim, and A. Reuther. High-productivity software development with pmatlab. *Computing in Science Engineering*, 11(1):75 –79, 2009.
 - [49] D. Marquez, J. Valera, A. Camelo, C. Aceros, M. Jimenez, and D. Rodriguez. Implementations of cyclic cross-ambiguity functions in fpgas for large scale signals. In *Circuits and Systems (LASCAS), 2011 IEEE Second Latin American Symposium on*, pages 1–4, Feb. 2011.

- [50] Rodríguez-López G. M. Toyos-González J. Mignucci-Giannoni, A. A. Summary of marine mammal strandings in puerto rico and the virgin islands. *Southeast U.S. Marine Mammal Stranding Network Workshop*, 1(5):100 – 124, 2000.
- [51] Servidio A. Martín, V. and S. García. Mass strandings of beaked whales in the canarian islands. *Proceedings of the Workshop on Active Sonar and Cetaceans (European Cetacean Society)*, 2(4):33 – 36, 2004.
- [52] Barlow J. Pitman-R. Balance L. Klinger-T. Taylor, B. A call for research to assess risk of acoustic impact on beaked whale populations. *IWC Scientific Commitee (SC/56/E36)*, 3(2), 2004.
- [53] Michael Jasny. The rising toll of sonar, shipping and industrial ocean noise on marine life. *Natural Resources Defense Council*, 1(1):1 – 84, 2005.