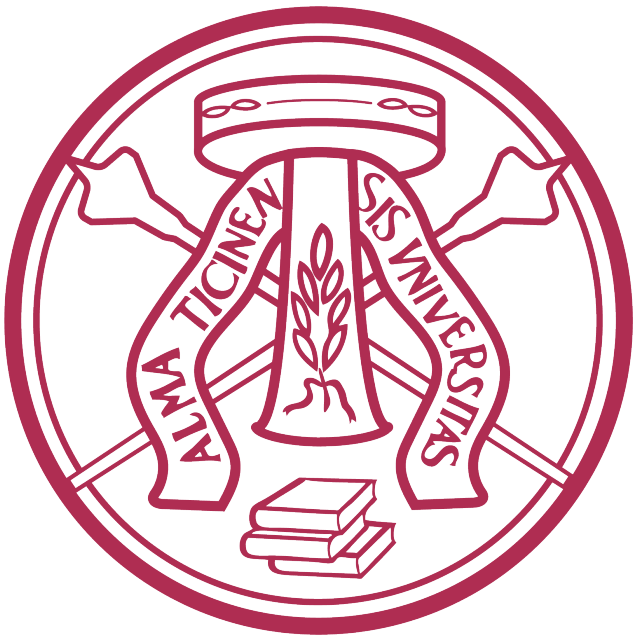

UNIVERSITY OF PAVIA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL, COMPUTER AND BIOMEDICAL ENGINEERING

MASTER'S DEGREE IN ELECTRONIC ENGINEERING

Deep learning for fractional channel estimation in OTFS receivers

Supervisor:

Prof. Pietro Savazzi

Co-supervisor:

Ing. Paolo Spallaccini

Candidate:

Mauro Marchese

A.Y. 2023/2024

Abstract

The rapid advancement of mobile communication technologies has necessitated innovative solutions for channel estimation, particularly in high-mobility scenarios where traditional methods struggle to maintain performance. This thesis addresses the challenges of fractional channel estimation in Orthogonal Time Frequency Space (OTFS) systems, a candidate waveform for 6G networks.

This work begins with a comprehensive review of wireless communication channels and existing channel estimation techniques for OTFS systems, highlighting their limitations in dealing with inter-path interference (IPI) in environments with fractional delays and Doppler shifts. To address these challenges, a variant of a state-of-the-art algorithm, named Progressive Inter-path Interference Cancellation (P-IPIC), is proposed to significantly reduce computational complexity and latency while improving estimation accuracy. Specifically, the results demonstrate that a single refinement procedure is sufficient to achieve good estimation performance.

Additionally, the integration of a deep learning (DL) approach into the channel estimation process is explored. A neural network model is developed to learn the intricate relationship between channel parameters and a corresponding parameter matrix. Simulation results show that while deep learning solutions contribute to latency reduction, they come with a minor, yet acceptable, performance loss.

The findings of this thesis offer valuable insights and practical solutions for next-generation wireless communication systems, particularly in the context of 6G technology.

Riassunto

Il rapido avanzamento delle tecnologie di comunicazione mobile ha reso necessarie soluzioni innovative per la stima del canale, in particolare in scenari ad alta mobilità, dove i metodi tradizionali faticano a mantenere le prestazioni. Questa tesi affronta le sfide della stima frazionaria del canale nei sistemi Orthogonal Time Frequency Space (OTFS), una forma d'onda candidata per le reti 6G.

Il lavoro inizia con una revisione completa dei canali di comunicazione wireless e delle tecniche di stima del canale esistenti per i sistemi OTFS, evidenziando le loro limitazioni nel gestire l'interferenza tra percorsi (IPI) in ambienti con ritardi e spostamenti Doppler frazionari. Per affrontare queste sfide, viene proposta una variante, denominata Progressive Inter-path Interference Cancellation (P-IPIC), di un algoritmo allo stato dell'arte, che mira a ridurre significativamente la complessità computazionale e la latenza, migliorando al contempo la precisione della stima. In particolare, i risultati dimostrano che un'unica procedura di affinamento è sufficiente per ottenere buone prestazioni di stima.

Inoltre, viene esplorata l'integrazione di un approccio di deep learning (DL) nel processo di stima del canale. Un modello di rete neurale viene sviluppato per apprendere la complessa relazione tra i parametri del canale e una matrice di parametri corrispondente. I risultati delle simulazioni mostrano che, sebbene le soluzioni basate sul deep learning contribuiscano alla riduzione della latenza, comportano una lieve, ma accettabile, perdita di prestazioni.

Le conclusioni di questa tesi offrono preziosi spunti e soluzioni pratiche per i sistemi di comunicazione wireless di prossima generazione, in particolare nel contesto della tecnologia 6G.

Contents

Acronyms

A/D Analog-to-digital. 25

ANN Artificial Neural Network. 55

AWGN Additive White Gaussian Noise. 29, 30

BER Bit Error Rate. 47, 51

CAF Cross-Ambiguity Function. 23

CDDPM Costituent Delay-Doppler Parameter Matrix. 34, 37, 43, 63, 64, 66, 68

CP Cyclic Prefix. 11

CP-FSK Continuous Phase Frequency Shift Keying. 10

CSI Channel State Information. 31, 32, 62

CSIR Channel State Information at Receiver. 31, 51

CUT Cell Under Test. 34

D/A Digital-to-analog. 25

DD Delay-Doppler. 21–25, 27, 29–31, 33–35, 46, 48, 63, 64

DDIPIC Delay-Doppler Inter-path Interference Cancellation. 32, 35, 37, 38, 40–45, 48, 50, 66, 68

DDRE Delay-Doppler Resource Element. 33

DFT Discrete Fourier Transform. 11, 27, 28

DL Deep Learning. 54, 55, 62, 63, 66, 68, 69

DPS Doppler Power Spectrum. 17, 46

DSSS Direct Sequence Spread Spectrum. 10

DZT Discrete Zak Transform. 25, 26, 28, 29, 32

FDMA Frequency Division Multiple Access. 10

FM Frequency Modulation. 10

FNN Feedforward Neural Network. 63

GMSK Gaussian Minimum Shift Keying. 10

GSM Global System for Mobile Communications. 10

I/O Input/output. 29

IDFT Inverse Discrete Fourier Transform. 27

IDZT Inverse Discrete Zak Transform. 25, 28

IoT Internet of Things. 54

IPI Inter-path Interference. 31–33, 35, 38, 40–43, 46, 48–51, 66, 68

IPIC Inter-path Interference Cancellation. 44

IQI Inphase Quadrature Imbalance. 62

ISAC Integrated Sensing and Communication. 69

ISFFT Inverse Symplectic Finite Fourier Transform. 23, 24, 27

ISI Intersymbol Interference. 11, 15

LMMSE Linear Minimum Mean Square Error. 47

M-MLE Modified Maximum Likelihood Estimator. 32

MIMO Multiple Input Multiple Output. 32, 54

MIP Multipath Intensity Profile. 15

ML Maximum Likelihood. 35, 47

NB Narrowband. 10, 16

NMSE Normalized Mean Square Error. 47, 48, 66

NRMSE Normalized Root Mean Square Error. 47, 48

OFDM Orthogonal Frequency Division Multiplexing. 11, 12, 19, 21, 22, 24, 25

OLS Ordinary Least Squares. 43

OMP Orthogonal Matching Pursuit. 62

OTFS Orthogonal Time Frequency Space. 12, 13, 19, 23–29, 31, 32, 45, 68

P-IPIC Progressive Inter-path Interference Cancellation. 41–45, 48, 50, 51, 66, 68, 69

PAM Pulse Amplitude Modulation. 25

PAPR Peak-to-average Power Ratio. 32

PDP Power Delay Profile. 15, 46

PMF Probability Mass Function. 47

PSD Power Spectral Density. 29, 47

PSNR Pilot Signal-to-noise Ratio. 34, 48, 49, 51, 66, 68

QAM Quadrature Amplitude Modulation. 23, 47

RCF Residue-based Cost Function. 40, 41, 44, 45

ReLU Rectified Linear Unit. 56, 63

RLS Regularized Least Squares. 43

RMS Root Mean Square. 15, 18

RX Receiver. 24

SBL Sparse Bayesian Learning. 32

SFFT Symplectic Finite Fourier Transform. 24–26, 28, 64

SFT Symplectic Fourier Transform. 22, 23

SNR Signal-to-noise Ratio. 47, 48, 51

SS Spread Spectrum. 10

TDL Tapped Delay Line. 19, 20

TDMA Time Division Multiple Access. 10

TF Time-frequency. 21, 22, 24, 26, 32, 63, 66, 68

TSE Two-Step Estimator. 32

TX Transmitter. 24

UE User Equipment. 46

UMTS Universal Mobile Telecommunication System. 10

WB Wideband. 10, 16

WCDMA Wideband Code Division Multiple Access. 10

Notation

Symbol	Description
a	Variable
$\mathbf{a}$	Column vector
$\mathbf{A}$	Matrix
$a[n]$	n-th element of vector $\mathbf{a}$
$A[n,k]$	(k,n)-th element of matrix $\mathbf{A}$
$(\cdot)^T$	Transposition
$(\cdot)^H$	Hermitian transposition
$\mathrm{diag}(\mathbf{a})$	Diagonal matrix with $\mathbf{a}$ as main diagonal
$\mathrm{vec}(\mathbf{A})$	Column-wise vectorization of the $M \times N$ matrix into a vector of length MN
$\mathrm{vec}^{-1}_{M,N}(\mathbf{a})$	Inverse vectorization of the vector $\mathbf{a}$ of length MN into a $M \times N$ matrix
$\mathbf{I}_N$	Identity matrix of order N
$\circledast$	Circular convolution
$\otimes$	Matrix Kronecker product
$[\cdot]_N$	Modulo-N operation
$\times$	Cartesian product of two sets

Chapter 1

Introduction and motivation

1.1 Mobile networks: a bit of history

The technological evolution of mobile wireless networks is characterized by the change in waveforms between different standards [1]. First-generation wireless networks employed analog waveforms to enable voice transmission. Frequency Division Multiple Access (FDMA) was used to enable spectrum sharing and resource allocation for multiple users, while Frequency Modulation (FM) was employed to encode information onto the transmitted signal.

In the early 1990s, the second generation (2G, often referred to as Global System for Mobile Communications (GSM)) began to take advantage of digital technology, allowing low-rate data communication in addition to voice services. Narrowband (NB) Gaussian Minimum Shift Keying (GMSK), a modulation technique belonging to the Continuous Phase Frequency Shift Keying (CP-FSK) family, was adopted. The GMSK waveform is characterized by a Gaussian phase-shaping filter with a Bandwidth-Time product of $BT = 0.3$, offering an optimal trade-off between spectral efficiency and phase distortion, with a modulation index of 0.5. Multiple users accessed the system through 200 kHz bandwidth channels, combined with Time Division Multiple Access (TDMA), enabling multiple users to communicate simultaneously using the same spectral resources. This marked the emergence of the concept of time-frequency resource blocks.

At the start of the new millennium, third-generation (3G) wireless systems were introduced. The Universal Mobile Telecommunication System (UMTS) standard employed Wideband (WB) technology for multiple access with channel bandwidth of 5 MHz. The well-known Wideband Code Division Multiple Access (WCDMA) is a Spread Spectrum (SS) technology that spreads information symbols over both time and frequency. Spreading is performed using Direct Sequence Spread Spectrum (DSSS), where the information sequence is multiplied by a higher-rate code sequence. Different users can be distinguished at the receiver if orthogonal codes are assigned to each

user. A well-known class of orthogonal codes is the Walsh-Hadamard codes.

The main drawback of the aforementioned technologies is that performance degrades as the data rate increases as a result of multipath effects. Multipath occurs when a signal is transmitted in a channel with various reflectors and scatterers. Any object that is not transparent to electromagnetic radiation reflects some energy, becoming a source of multipath interference. At low data rates, the symbol duration is longer than the delay spread, making the wireless channel effectively flat in frequency over the desired bandwidth. However, when data rates increase, the symbol duration shortens, and the delay spread becomes greater than the symbol time. This leads to the well-known phenomenon of Intersymbol Interference (ISI), making the wireless channel frequency-selective.

To address this, multicarrier modulations have been developed to perform well in frequency-selective channels. The basic idea behind multicarrier modulation is to divide the channel into narrowband subchannels, where the channel response is nearly flat. These studies led to the development of Orthogonal Frequency Division Multiplexing (OFDM), which is the predominant waveform used in 4G and 5G standards. The main advantage of OFDM systems is the near-optimal performance in multipath scenarios together with low-complexity implementation through Discrete Fourier Transform (DFT) and single-tap equalization technique, enabled by the usage of a Cyclic Prefix (CP).

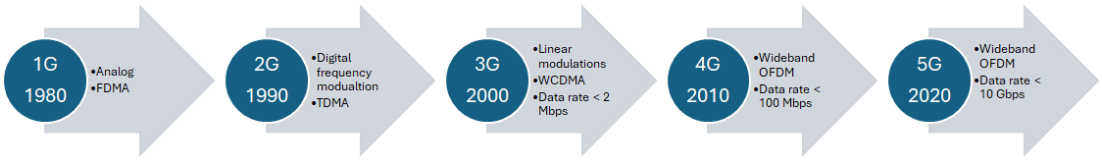

Figure 1.1: The evolution of mobile networks.

1.2 ”What comes next?”

As depicted in [2, 3], one of the main limitations of current standards is their performance in high-mobility scenarios. OFDM offers near optimal performance in static channels; however, when users and reflecting objects are moving, the channel becomes time-varying, and the Doppler effect destroys the orthogonality of the subchannels. The 6G technology must address new use cases, where high-speed trains, drones, self-driving cars, and other moving objects are communicating with each other. As shown

in Fig 1.2, while the maximum speed supported by 5G is 500 km/h, 6G aims to support speeds up to 1000 km/h. Since each generation is defined by its waveform, and OFDM is no longer sufficient in high-mobility scenarios, new waveforms have being explored and studied to meet these demands.

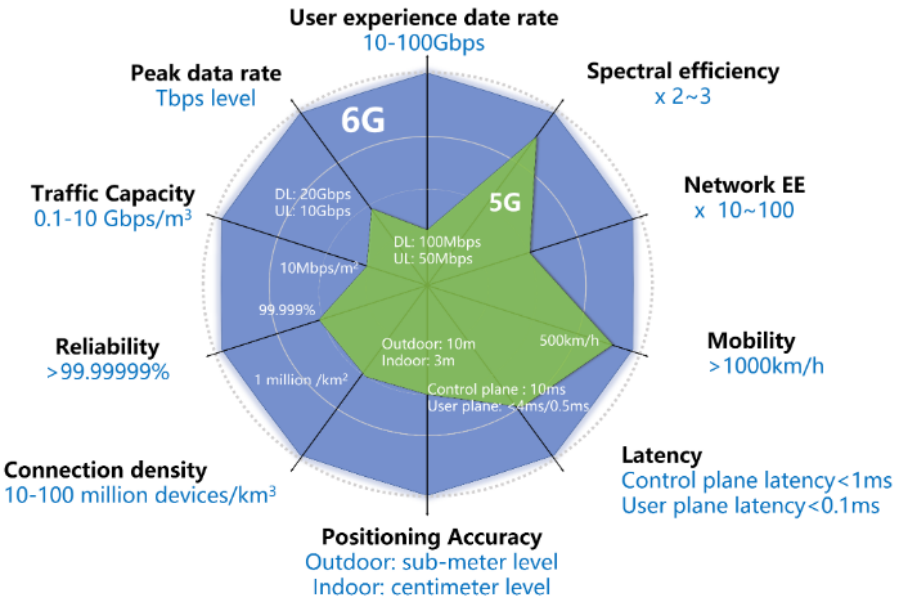

Figure 1.2: Comparison of 5G and 6G systems [4].

In particular, in [5] an overview of different waveforms for high mobility scenarios is reported. The main idea is to exploit intrinsic properties of different domains such as time-frequency, delay-Doppler, delay-scale and delay-sequency. Different transforms are used to exchange domains (Mellin, Walsh-Hadamard, discrete affine Fourier, ...). In the following we are going to focus on Orthogonal Time Frequency Space (OTFS) modulation that is considerate as the candidate waveform for 6G networks. In particular, the channel estimation problem in practical environment, where channel parameters may assume fractional values, is faced.

1.3 Objectives

As mentioned before, the scope of this thesis is channel estimation when fractional channel parameters are assumed. Hence, due to time and bandwidth limitations the resolution is not enough and off-grid values must be considered.

The main objective oh this thesis are

1. *Low- latency/complexity solution*: the first part of the work is devoted to simplify

the operation of an already existing algorithm in order to reduce the computational burden,

2. *Deep learning based solution*: in the second part the idea is to exploit deep learning in order to further reduce the complexity by learning a one-to-one relation used to build a parameter matrix.

The rest of the thesis is organized as follows: preliminaries on wireless communications are discussed, OTFS modulation is introduced and the channel estimation problem is formulated. After that, the state of the art is presented and the proposed approach is discussed.

Chapter 2

High-mobility channels

2.1 Preliminaries on wireless channels

In wireless communication systems, the transmission medium between the transmitter and the receiver is known as the wireless channel. Unlike wired channels, which are typically more predictable, wireless channels introduce several complexities due to the nature of the environment in which signals propagate. These complexities arise from factors such as the mobility of users, environmental obstacles, and the inherent properties of the radio waves themselves. Understanding the characteristics and impairments of wireless channels is crucial for the design and optimization of reliable communication systems.

Wireless communication is inherently affected by the environment in which signals propagate. Among the various challenges, multipath propagation and mobility stand out as two of the most critical factors influencing the performance of wireless systems.

In the following sections, the goal is to build the baseband equivalent model of the wireless channel according to [1].

2.1.1 Multipath

Multipath propagation occurs when the transmitted signal takes multiple paths to reach the receiver. These paths arise due to reflections, diffractions, and scattering caused by physical objects in the environment, such as buildings, trees, vehicles, and even the ground. Each of these paths has a different length, causing the signal copies to arrive at the receiver with varying delays and phases.

Denoting $s(t)$ the baseband equivalent of the transmitted signal, the received signal is given as

$$r(t) = \sum_{i=1}^{P} g_i s(t - \tau_i) e^{-j2\pi f_c \tau_i}, \tag{2.1}$$

where P is the number of propagation paths, g_i is the complex gain associated with the i-th path, τ_i is the propagation delay of the i-th received copy and f_c is the carrier frequency. Typically, wireless environment are characterized by their Power Delay Profile

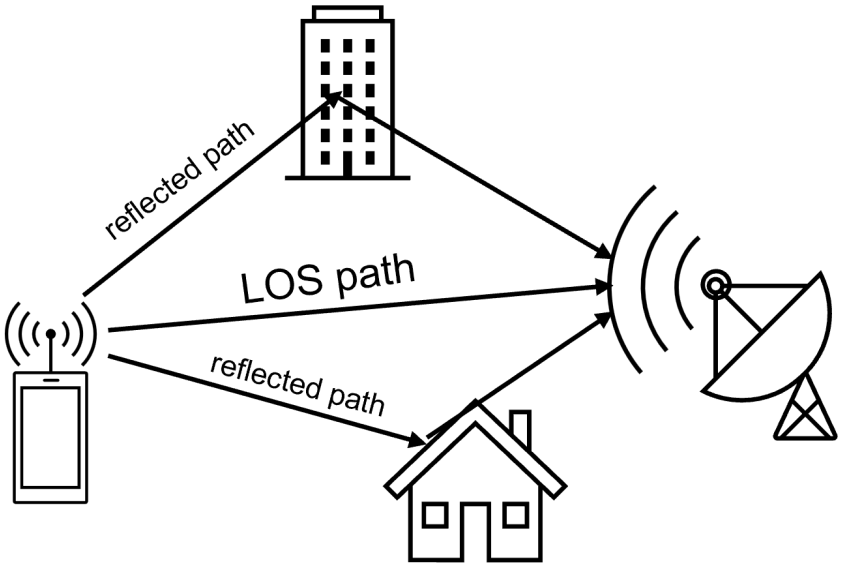

Figure 2.1: Multipath effect.

(PDP) or Multipath Intensity Profile (MIP). This function, denoted $P(\tau)$, provides the average power at the output of the channel for a given propagation delay and is used to extract statistics of a given channel. The PDP is usually extracted by measurements.

This time dispersion experienced in a multipath environment results in what is known as delay spread. A common definition of delay spread is given by the Root Mean Square (RMS) value as

$$\sigma_\tau = \sqrt{\frac{\int_0^{+\infty}(\tau-\mu_\tau)^2 P(\tau)d\tau}{\int_0^{+\infty} P(\tau)d\tau}}, \tag{2.2}$$

where μ_τ is the average delay and it is defined as

$$\mu_\tau = \frac{\int_0^{+\infty}\tau P(\tau)d\tau}{\int_0^{+\infty} P(\tau)d\tau}. \tag{2.3}$$

With this definition, we can immediately understand that if the delay spread is much higher than the symbol duration, ISI will occur. On the other hand, if the symbol duration is much higher than the delay spread, multipath can be neglected.

The time-invariant impulse response of the channel is given by

$$g(\tau) = \sum_{i=1}^{P} g_i \delta(\tau - \tau_i) e^{-j2\pi f_c \tau_i}. \tag{2.4}$$

Since the path gain is complex, the phase rotation due to the propagation delay can be included in the channel gain leading to the following expression

$$g(\tau) = \sum_{i=1}^{P} g_i \delta(\tau - \tau_i). \tag{2.5}$$

Alternatively, it is possible to analyze the channel in the frequency domain using the Fourier transform. Thus, the frequency response of the channel is

$$H(f) = \mathcal{F}\{g(\tau)\} = \sum_{i=1}^{P} g_i e^{-j2\pi f \tau_i}. \tag{2.6}$$

In the frequency domain, it is possible to define the channel coherence bandwidth as the range of frequencies over which the channel is almost flat. As a rule of thumb, the channel coherence bandwidth can be obtained as

$$(\Delta f)_c \approx \frac{1}{\sigma_\tau}. \tag{2.7}$$

With this definition, the channel is frequency non-selective (or flat) if the signal bandwidth is lower than the coherence bandwidth of the channel. In this case, the channel is said to be NB. On the contrary, the channel is frequency selective if the signal bandwidth is higher than the coherence bandwidth of the channel and the channel is said to be WB. A summary of the properties of the multipath channel is reported in Table 2.1, where T denotes the symbol time and B the signal bandwidth.

Channel	Time	Frequency
Frequency non-selective	$\sigma_\tau \ll T$	$(\Delta f)_c \gg B$
Frequency selective	$\sigma_\tau \gg T$	$(\Delta f)_c \ll B$

Table 2.1: Multipath channel characterization.

2.1.2 Mobility

In mobile environments, both the transmitter and receiver, or objects in the surroundings, can be in motion. This mobility introduces an additional layer of complexity to the wireless channel. The most prominent effect caused by mobility is the Doppler

shift. The frequency shift caused by Doppler effect due to mobility in the i-th path is given by

$$\nu_i = \frac{v_i \cos\theta_i}{c} f_c, \tag{2.8}$$

where v_i is the speed of the moving object, θ_i is the angle-of-arrival of the received copy and c is the speed of light. Denoting $s(t)$ the baseband equivalent of the trans-

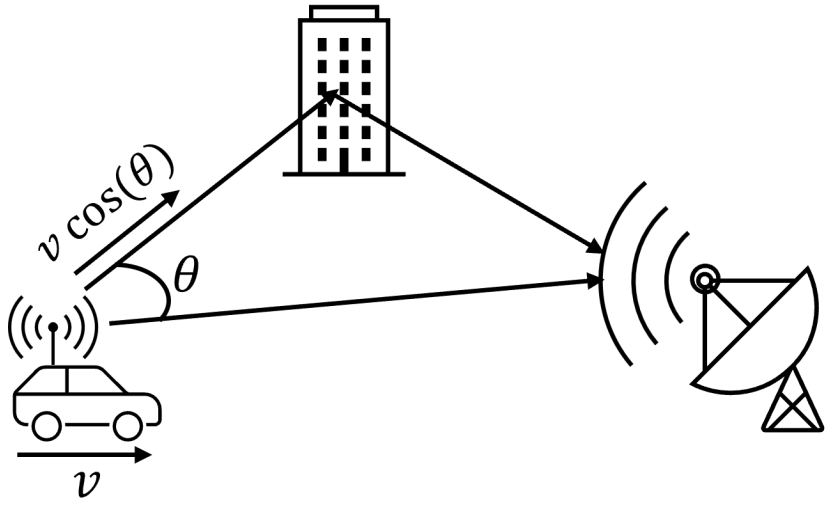

Figure 2.2: High-mobility channel.

mitted signal, the received signal after passing through a channel affected by multipath and mobility is given as

$$r(t) = \sum_{i=1}^{P} g_i s(t-\tau_i) e^{j2\pi\nu_i(t-\tau_i)}. \tag{2.9}$$

Thus the channel is characterized by the time-variant impulse response

$$g(t,\tau) = \sum_{i=1}^{P} g_i \delta(\tau-\tau_i) e^{j2\pi\nu_i(t-\tau_i)}. \tag{2.10}$$

Similarly to multipath, mobility is characterized by the Doppler Power Spectrum (DPS). This function, denoted $P(\nu)$, provides the average power at the output of the channel for a given Doppler shift. The DPS is usually defined by a model. A very well-known model is given by the Jakes' model where it is assumed that the angle of arrival is uniformly distributed.

The frequency dispersion caused by the results of the Doppler effect is measured through the Doppler spread. As for the delay spread, a common definition of Doppler

spread is given by the RMS value as

$$\sigma_\nu = \sqrt{\frac{\int_{-\infty}^{+\infty}(\nu-\mu_\nu)^2 P(\nu)d\nu}{\int_{-\infty}^{+\infty}P(\nu)d\nu}}, \tag{2.11}$$

where μ_ν is the average Doppler shift and it is defined as

$$\mu_\nu = \frac{\int_{-\infty}^{+\infty}\nu P(\nu)d\nu}{\int_{-\infty}^{+\infty}P(\nu)d\nu}. \tag{2.12}$$

In the time domain, it is possible to define the channel coherence time as the time duration over which the channel is almost constant. As a rule of thumb, the channel coherence time can be obtained as

$$(\Delta t)_c \approx \frac{1}{\sigma_\nu}. \tag{2.13}$$

With this definition, the channel is slow-varying if the symbol time is much lower than the coherence time of the channel. In this case, the channel is almost static. On the contrary, the channel is fast varying if the symbol time is higher than the coherence time of the channel. Table 2.2 provides a summary of this distinction.

Channel	Time
Slow varying	$T \ll (\Delta t)_c$
Fast varying	$T \gg (\Delta t)_c$

Table 2.2: Channel behavior depending on mobility.

2.2 Continuous-time channel model

In the previous section the two main channel impairments, multipath and mobility, were discussed. Based on that discussion, it is possible to develop the continuous-time channel model.

In real systems, a receiver has finite delay and Doppler resolutions due to limited bandwidth and time duration. In a multicarrier system, if the number of subcarriers is M and the subcarrier spacing is Δf the signal bandwidth is given by $M\Delta f$. Thus the delay resolution is given by $\tau_{\text{res}} = \frac{1}{M\Delta f}$. Similarly, if M symbols are transmitted in a single time slot of duration T and a frame is made of N time slots, the total signal duration is NT. Thus, the Doppler resolution is given by $\nu_{\text{res}} = \frac{1}{NT}$. Due to this limitation, the receiver cannot see the true values of the channel parameters. Hence,

assuming sufficient resolution, true channel parameters can be replaced with on-grid values

$$\tau_i = \frac{\ell_i}{M\Delta f}, \nu_i = \frac{\kappa_i}{NT}, \tag{2.14}$$

where $\ell_i \in \mathbb{Z}^+$ and $\kappa_i \in \mathbb{Z}$ are the normalized delay and Doppler shift, respectively.

At this point, it is important to mark that the subcarrier spacing and the time slot duration are related by $\Delta f = \frac{1}{T}$. This is done to ensure the orthogonality of the basis functions and it is used in both OFDM and OTFS systems.

With these definitions, the time-varying impulse response of the channel can be written as

$$g(t,\tau) = \sum_{\ell\in\mathcal{L}} \sum_{\kappa\in\mathcal{K}_\ell} \mathrm{v}_\ell(\kappa)\delta\Big(\tau - \frac{\ell}{M\Delta f}\Big) e^{j2\pi\frac{\kappa}{NT}\left(t-\frac{\ell}{M\Delta f}\right)}. \tag{2.15}$$

where $\mathcal{L}$ is the set of normalized delays and $\mathcal{K}_\ell$ is the set of normalized Doppler shifts at delay ℓ. Moreover, $\mathrm{v}_\ell(\kappa)$ is called Doppler response at delay ℓ and is given by

$$\mathrm{v}_\ell(\kappa) = \begin{cases} g_i & \text{if } \ell = \ell_i, \kappa = \kappa_i, \\ 0 & \text{otherwise.} \end{cases} \tag{2.16}$$

2.3 Discrete-time channel model

In order to study and analyze waveforms, it is convenient to use a discrete-time model since at receiver side the down-converted signal is then sampled at the symbol rate $f_s = B = M\Delta f$. Hence, MN samples per frame are produced.

The received signal is obtained computing the time-variant convolution integral

$$r(t) = \int_{-\infty}^{+\infty} g(t,\tau)s(t-\tau)d\tau. \tag{2.17}$$

After sampling, the received signal is obtained as

$$r[q] = r\Big(q\frac{T}{M}\Big) = \sum_{l=0}^{M-1} g^s\left[l,q\right] s\left[q-l\right], \tag{2.18}$$

where $g^s\left[l,q\right] = g\Big(\frac{qT}{M}, \frac{l}{M\Delta f}\Big)$ is the discrete-time impulse response and is given by

$$g^s[l,q] = \sum_{\ell\in\mathcal{L}} \sum_{\kappa\in\mathcal{K}_\ell} \mathrm{v}_\ell(\kappa)\delta[l-\ell]z^{\kappa(q-l)} = \begin{cases} \sum_{\kappa\in\mathcal{K}_\ell} \mathrm{v}_\ell(\kappa)z^{\kappa(q-\ell)} & \text{if } l = \ell \in \mathcal{L}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.19}$$

and $z = e^{j\frac{2\pi}{MN}}$. Figure 2.3 shows the Tapped Delay Line (TDL) model for the linear

time-invariant channel, while Figure 2.4 shows the TDL model for the time-varying channel.

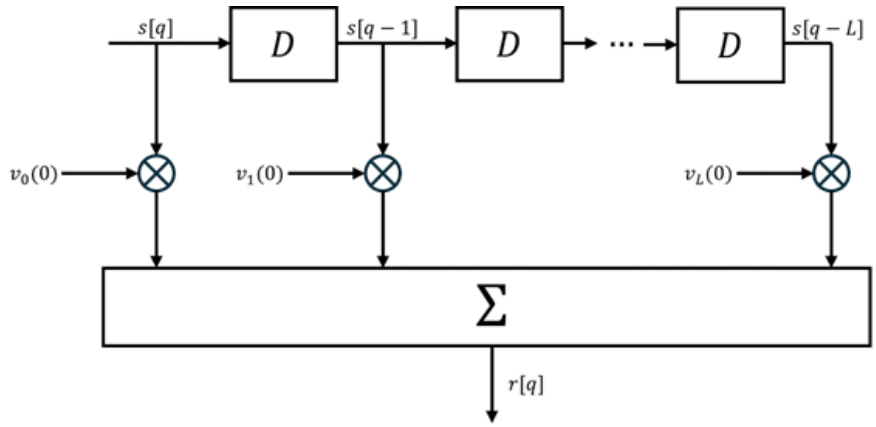

Figure 2.3: TDL model for a static multipath channel.

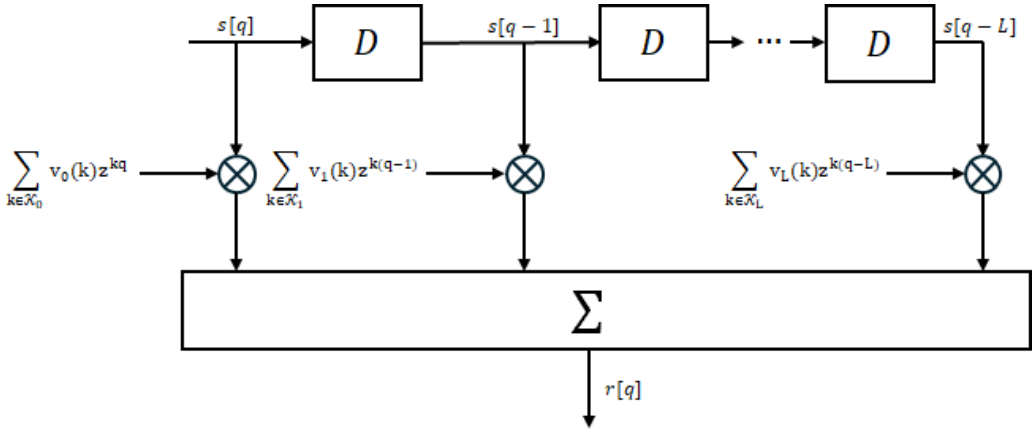

Figure 2.4: TDL model for a high-mobility multipath channel.

Chapter 3

Orthogonal Time Frequency Space

3.1 Delay-Doppler domain

Orthogonal time frequency space was first introduced in [6, 7] and it exploits Delay-Doppler (DD) domain features to carry information over a rapidly varying channel due to high Doppler effects.

Before starting the discussion on OTFS, it is worth studying the DD domain and understanding its characteristics and relations with other domains.

OFDM multiplexes information symbols in the Time-frequency (TF) domain where both time and frequency selectivity are experienced. The wireless channel in the time-frequency domain is obtained taking the Fourier Transform along the delay dimension of the time-varying impulse response as

$$H(f,t) = \int_{-\infty}^{+\infty} g(t,\tau) e^{-j2\pi f\tau} d\tau. \tag{3.1}$$

Figure 3.1 shows an example of the TF wireless channel.

The channel can be represented in the delay-Doppler domain taking the Fourier transform along the time dimension

$$h(\tau,\nu) = \int_{-\infty}^{+\infty} g(t,\tau) e^{-j2\pi\nu(t-\tau)} dt = \sum_{i=1}^{P} g_i \delta(\tau-\tau_i)\delta(\nu-\nu_i). \tag{3.2}$$

It is interesting to notice that the wireless channel becomes *sparse* in the DD domain. In particular, the channel is made of few peaks (one for every path) that are placed in the DD domain according to the delay and Doppler coordinates. Moreover, the channel is *compact* since only $3P$ parameters are needed to characterize the channel (channel

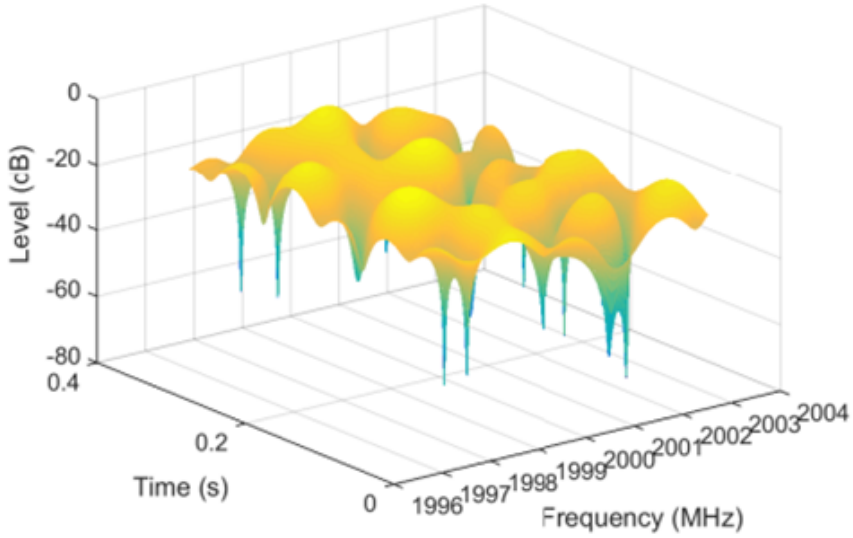

Figure 3.1: Wireless channel in the time-frequency domain.

coefficients, delays and Doppler shifts for each path). In the end, we can notice that the channel is *time invariant* within a geometric coherence window of the channel, over which we can assume that the physical geometry of the channel is unchanged. Since this time and typical transmission time are usually much higher than the coherence time of the channel, this suggests that delay-Doppler domain communication is suitable to overcome OFDM limitations.

The received signal can be obtained from the delay-Doppler channel response through an Heisenberg transform as

$$r(t) = \iint h(\tau, \nu) s(t-\tau) e^{j2\pi\nu(t-\tau)} d\tau d\nu. \tag{3.3}$$

In order to complete the overview of the relations among different domains, the relation between the delay-Doppler domain and the time-frequency domain must also be provided. Once the TF representation of the channel is known, the DD channel response is obtained via Symplectic Fourier Transform (SFT) as

$$h(\tau, \nu) = \text{SFT}\{H(f,t)\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} H(f,t) e^{-j2\pi(\nu t - f\tau)} df dt. \tag{3.4}$$

Conversely, one can pass from he DD representation to the time-frequency via inverse

SFT as

$$H(f,t) = \text{ISFT}\{h(\tau,\nu)\} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} h(\tau,\nu)e^{j2\pi(\nu t - f\tau)}d\tau d\nu. \tag{3.5}$$

3.2 System model

In this section, the OTFS system is presented and modulator and demodulator architectures are discussed. Both two-step architecture based on SFT and Heisenberg transform and one-step architecture based on Zak transform are discussed. Moreover, matrix formulation for OTFS is introduced since it is useful for analysis and simulation and will be used to formulate the channel estimation problem.

3.2.1 Modulation

OTFS multiplexes MN information symbols taken from an alphabet (e.g. Quadrature Amplitude Modulation (QAM) constellation) in the delay-Doppler grid referred to as $I_{DD} = \{(m\tau_{\text{res}}, n\nu_{\text{res}}),\ m = 0,1,...,M-1,\ n = 0,1,...,N-1\}$.

The DD-domain symbols are then converted to the time-frequency domain through Inverse Symplectic Finite Fourier Transform (ISFFT). Denoting with $\mathbf{X} \in \mathbb{C}^{M\times N}$ the delay-Doppler symbols multiplexed in the DD domain, the corresponding samples in the time-frequency domain are obtained as

$$X_{tf}[l,k] = \frac{1}{\sqrt{MN}}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} X[m,n]e^{j2\pi\left(\frac{nk}{N}-\frac{ml}{M}\right)}. \tag{3.6}$$

After that, the time-domain OTFS signal is obtained through the Heisenberg transform as

$$s(t) = \sum_{k=0}^{N-1}\sum_{l=0}^{M-1} X_{tf}[l,k]g_{tx}(t-kT)e^{j2\pi l\Delta f(t-kT)}, \tag{3.7}$$

where $g_{tx}(t)$ is the transmit shaping pulse of duration T.

3.2.2 Demodulation

At the receiver side, inverse operations are performed to recover the received symbols in the DD-domain. Firstly, the received signal is passed through a matched filter computing the so-called Cross-Ambiguity Function (CAF) $A_{g_{rx},r}(f,t)$ to convert the time-domain signal into the time-frequency representation

$$Y_{tf}(f,t) = A_{g_{rx},r}(f,t) = \int r(t')g^*_{rx}(t'-t)e^{-j2\pi f(t'-t)}dt'. \tag{3.8}$$

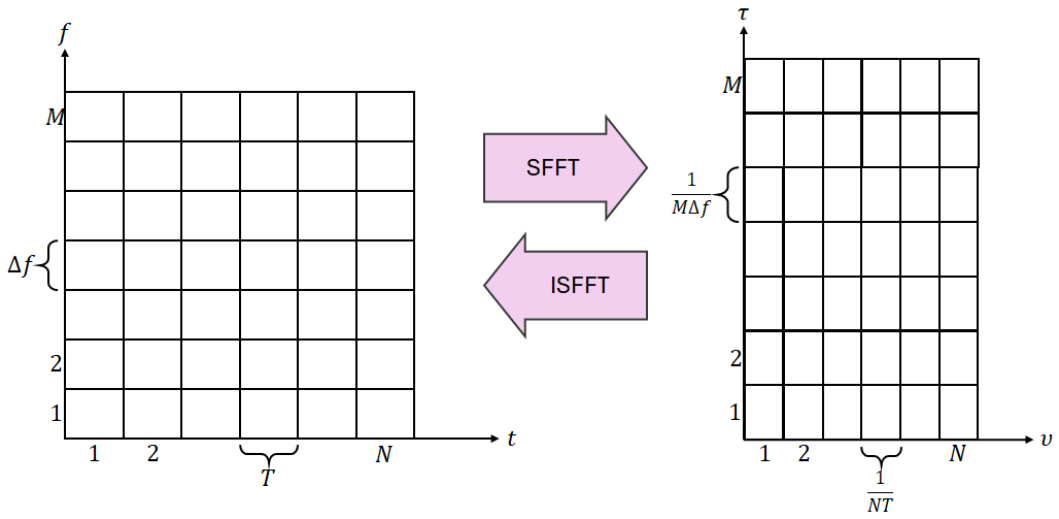

Figure 3.2: TF and DD grids with corresponding resolutions.

Notice that $g_{rx}(t)$ denotes the receive shaping pulse. After that, time-frequency symbols are obtained performing two-dimensional sampling at the TF grid points

$$Y_{tf}[l,k] = Y_{tf}(f,t)|_{f=l\Delta f, t=kT}. \tag{3.9}$$

Secondly, a Symplectic Finite Fourier Transform (SFFT) decoder is applied to get the DD-domain received symbols

$$Y[m,n] = \frac{1}{\sqrt{MN}} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} Y_{tf}[l,k] e^{-j2\pi\left(\frac{nk}{N} - \frac{ml}{M}\right)}. \tag{3.10}$$

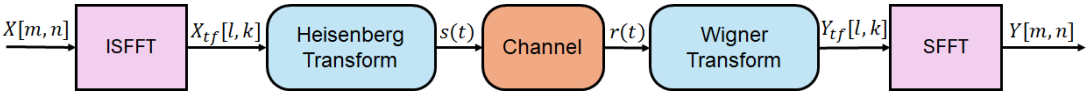

Figure 3.3: Two-step OTFS system architecture.

3.2.3 Two-step system architecture

Figure 3.3 shows the architecture of the OTFS system as first introduced in the literature. It is based on time-frequency modulator and demodulator and precoding and combining in the delay-Doppler domain based on SFFT.

The two-step architecture emphasizes the compatibility of OTFS with 4G and 5G systems based on OFDM. In particular, OTFS can be seen as a 2D precoding based on the ISFFT at the Transmitter (TX). After that, at the Receiver (RX), a post-processing

SFFT decoder can be used to recover symbols in the DD domain. The main difference is that the time-frequency modulator operates on N OFDM symbols instead of processing one OFDM symbol at a time.

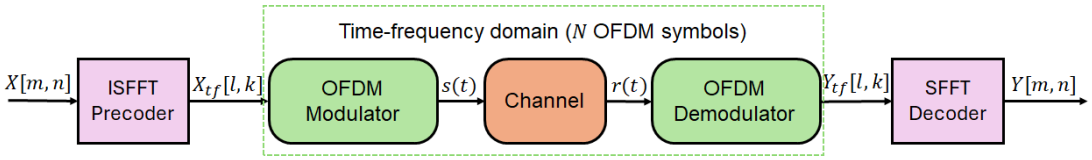

Figure 3.4: Compatibility with OFDM systems.

3.2.4 DZT-based OTFS system

In the OTFS literature it is shown that OTFS can be implemented using the Zak transform, which allows direct transformation of the delay-Doppler symbols into a time-domain signal. The architecture of the single-step OTFS system si shown in Figure 3.5, where the Inverse Discrete Zak Transform (IDZT) is used to convert the delay-Doppler information symbols into a discrete-time signal as

$$s[q] = s[m + nM] = \frac{1}{\sqrt{N}} \sum_{p=0}^{N-1} X[m, p] e^{j2\pi \frac{np}{N}}. \tag{3.11}$$

Then, a Digital-to-analog (D/A) converter is used to obtain the continuous-time transmitted signal $s(t)$.

At the receiver, an Analog-to-digital (A/D) converter is used to obtain the discrete-time observations $r[q] = r[m+pM]$ from $r(t)$ and, after that, a Discrete Zak Transform (DZT) is used to recover the DD domain information symbols as

$$Y[m, n] = \frac{1}{\sqrt{N}} \sum_{p=0}^{N-1} r[m + pM] e^{-j2\pi \frac{np}{N}}. \tag{3.12}$$

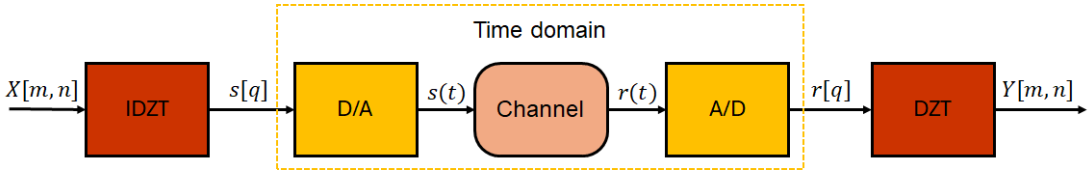

Figure 3.5: Single-step OTFS system architecture based on Zak transform.

A more general model for the D/A and A/D conversions taking into account a real interpolation pulse is given by Pulse Amplitude Modulation (PAM). In this case the

transmitted signal is written as

$$s(t) = \sum_{q=0}^{MN-1} s[q]p(t - qT). \tag{3.13}$$

At the receiver, discrete-time observations are obtained using a matched filter $p^*(-t)$ followed by a sampler as

$$r[q] = \int_{-\infty}^{+\infty} r(t)p^*(t' - t)dt' \Bigg|_{t=\frac{qT}{M}}. \tag{3.14}$$

The equality between the two-step architecture and the DZT-based on will be more clear in the section covering the matrix formulation for OTFS.

3.2.5 Biorthogonality condition

Before discussing real implementations of OTFS with practical pulse shaping waveforms it is worth to spend a few words about the case of ideal pulses. The transmit and receive pulses are said to be biorthogonal if

$$A_{g_{tx},g_{rx}}(f, t) = \delta(t)\delta(f). \tag{3.15}$$

If biorthogonality is reached, the received time-frequency samples would not suffer from intersymbol interference. Thus, after sampling, the matched filter output becomes

$$Y_{tf}[l, k] = H_{tf}[l, k]X_{tf}[l, k], \tag{3.16}$$

where

$$H_{tf}[l, k] = H(f, t)|_{f=l\Delta f,t=kT} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(\tau, \nu)e^{j2\pi(\nu kT - l\Delta f\tau)}d\tau d\nu. \tag{3.17}$$

Hence, single-tap equalization can be used in the time-frequency domain to compensate for channel distortion.

In order to achieve this condition we would like to have a pulse that is perfectly localized in both time and frequency at the output of the matched filter. Unfortunately, due to the Heisenberg uncertainty principle, it is impossible to have real pulses that satisfy biorthogonality condition being perfectly localized in the TF plane. Only approximate biorthogonality can be achieved with a proper pulse-shaping design.

As a benchmark, it is interesting to understand how OTFS works in the ideal case of biorthogonal pulses. In this case, we can apply a SFFT to get the DD received symbols. Exploiting SFFT properties we obtain that the received DD symbols are given by the

2D circular convolution

$$Y[m,n] = H_{dd}[m,n] \circledast X[m,n] = \sum_{i=1}^{P} g_i X\big[[m-l_i]_M, [n-k_i]_N\big], \tag{3.18}$$

where $\mathbf{H}_{dd} = \text{SFFT}\{\mathbf{H}_{tf}\}$. This suggest that if a symbol is transmitted in the DD domain, at the receiver P copies of that symbol are observed at P different coordinates in the DD domain. In other words, also in the ideal scenario, equalization in the DD domain is more complex than in the time-frequency.

3.3 Matrix formulation

As discussed above, a compact matrix formulation for OTFS is valuable for analysis and simulation purposes. In this section, the matrix formulation is introduced. A more general matrix formulation taking into account the effect of fractional channel parameters will be introduced in Chapter 4.

3.3.1 Transmitter

In the previous section, OTFS was shown to be implemented using an ISFFT precoder. This operation is performed by means of a direct M-point discrete Fourier transform (DFT) along delay (columns of $\mathbf{X}$) and an inverse N-point DFT along Doppler (rows of $\mathbf{X}$). Thus we can write

$$\mathbf{X}_{tf} = \mathbf{F}_M \mathbf{X} \mathbf{F}_N^H, \tag{3.19}$$

where $\mathbf{F}_N$ denotes the unitary DFT matrix and is given as

$$\mathbf{F}_N = \frac{1}{\sqrt{N}} \big\{ e^{j2\pi \frac{mq}{N}} \big\}_{m,q=0}^{N-1}. \tag{3.20}$$

Notice that the inverse DFT is simply implemented by means of the Hermitian cojugate operation.

Time-frequency samples are then converted to delay-time samples via M-point Inverse Discrete Fourier Transform (IDFT) along frequency (columns of $\mathbf{X}_{tf}$)

$$\tilde{\mathbf{X}} = \mathbf{F}_M^H \mathbf{X}_{tf} = \mathbf{F}_M^H \mathbf{F}_M \mathbf{X} \mathbf{F}_N^H = \mathbf{X} \mathbf{F}_N^H. \tag{3.21}$$

After that, pulse shaping is performed multiplying the delay-time matrix $\tilde{\mathbf{X}}$ by the following pulse shaping matrix

$$\mathbf{G}_{tx} = \text{diag}\left\{ g_{tx}(0), g_{tx}\Big(\frac{T}{M}\Big), ..., g_{tx}\Big(\frac{(M-1)T}{M}\Big) \right\}. \tag{3.22}$$

In the end, the time-domain samples are obtained performing column-wise vectorization. Thus, the Heisenberg transform modulator in matrix form is given as

$$\mathbf{s} = \mathrm{vec}\big(\mathbf{G}_{tx}\tilde{\mathbf{X}}\big) = \mathrm{vec}\big(\mathbf{G}_{tx}\mathbf{F}_M^H\mathbf{X}_{tf}\big). \tag{3.23}$$

It can be noticed that, for rectangular pulse shaping ($\mathbf{G}_{tx} = \mathbf{I}_M$ is the identity matrix of order M), the transmitted samples are given by the IDZT as

$$\mathbf{s} = \mathrm{vec}\big(\mathbf{X}\mathbf{F}_N^H\big). \tag{3.24}$$

Figure 3.6 shows the implementation of the Zak-transform based transmitter.

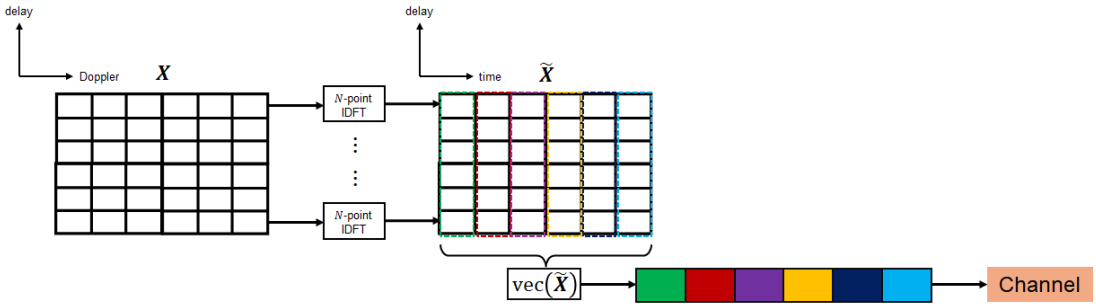

Figure 3.6: DZT-based OTFS transmitter.

3.3.2 Receiver

At the receiver, inverse operations are performed. The received samples vector is converted to the delay-time domain by means of a reshaping into a matrix and applying the receive pulse shaping $\mathbf{G}_{rx}$. Thus the received delay-time samples matrix is given as

$$\tilde{\mathbf{Y}} = \mathbf{G}_{rx}\mathrm{vec}_{M,N}^{-1}(\mathbf{r}). \tag{3.25}$$

After that, time-frequency samples are obtained via M-point DFT along delay (columns of $\tilde{\mathbf{Y}}$) as

$$\mathbf{Y}_{tf} = \mathbf{F}_M\tilde{\mathbf{Y}}. \tag{3.26}$$

Thus the Wigner transform demodulator in matrix form is given as

$$\mathbf{Y}_{tf} = \mathbf{F}_M\mathbf{G}_{rx}\mathrm{vec}_{M,N}^{-1}(\mathbf{r}). \tag{3.27}$$

Finally, a SFFT is applied to obtain the received information symbols matrix as

$$\mathbf{Y} = \mathbf{F}_M^H\mathbf{Y}_{tf}\mathbf{F}_N. \tag{3.28}$$

Moreover, in the case of rectangular pulse shaping ($\mathbf{G}_{rx} = \mathbf{I}_M$), the received DD symbols matrix is given by the DZT as

$$\mathbf{Y} = \mathrm{vec}_{M,N}^{-1}(\mathbf{r})\mathbf{F}_N. \tag{3.29}$$

Figure 3.7 shows the implementation of the Zak-transform based receiver.

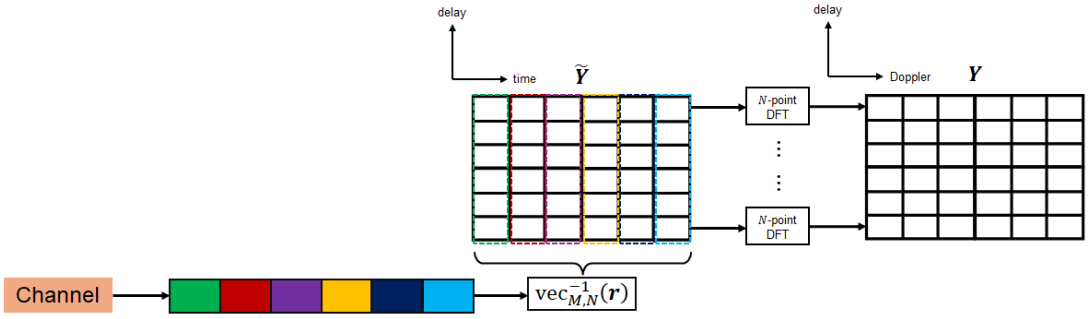

Figure 3.7: DZT-based OTFS receiver.

3.3.3 Relations in vector form

Since the goal is to express the input-output relations in matrix form it is more convenient to use vectorized relations. The vectorized transmitted and received DD information symbol matrices are given as $\mathbf{x} = \mathrm{vec}(\mathbf{X})$ and $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$, respectively.

With this definitions, modulation is performed as

$$\mathbf{s} = \left(\mathbf{F}_N^H \otimes \mathbf{G}_{tx}\right)\mathbf{x}. \tag{3.30}$$

At the receiver, demodulation is performed as

$$\mathbf{y} = \left(\mathbf{F}_N \otimes \mathbf{G}_{rx}\right)\mathbf{r}. \tag{3.31}$$

3.3.4 Input-output relations

At this point it is possible to formulate input-output relations starting from the Input/output (I/O) relation in the time-domain. As explained previously, the received signal is given by the convolution sum in Equation (2.18). This relation can be put in matrix form as

$$\mathbf{r} = \mathbf{G}\mathbf{s} + \mathbf{w}, \tag{3.32}$$

where $\mathbf{w} \sim \mathcal{CN}(0, N_0\mathbf{I}) \in \mathbb{C}^{MN\times 1}$ denotes Additive White Gaussian Noise (AWGN) with monolateral Power Spectral Density (PSD) N_0 and $\mathbf{G} \in \mathbb{C}^{MN\times MN}$ is the time-

domain channel matrix given as

$$G[q, q-l] = \begin{cases} g^s[l, q] & q \geq l, \\ 0 & \text{otherwise.} \end{cases} \tag{3.33}$$

In order to derive the input-output relation in the delay-Doppler domain one can multiply both sides of Equation (3.32) by $\mathbf{F}_N \otimes \mathbf{G}_{rx}$. Thus, we get

$$\big(\mathbf{F}_N \otimes \mathbf{G}_{rx}\big)\mathbf{r} = \big(\mathbf{F}_N \otimes \mathbf{G}_{rx}\big)\mathbf{G}\mathbf{s} + \big(\mathbf{F}_N \otimes \mathbf{G}_{rx}\big)\mathbf{w}. \tag{3.34}$$

Denoting with $\mathbf{z} = \big(\mathbf{F}_N \otimes \mathbf{G}_{rx}\big)\mathbf{w}$ the AWGN noise vector in the DD domain and substituting Equation (3.30) we get

$$\mathbf{y} = \big(\mathbf{F}_N \otimes \mathbf{G}_{rx}\big)\mathbf{G}\big(\mathbf{F}_N^H \otimes \mathbf{G}_{tx}\big)\mathbf{x} + \mathbf{z}. \tag{3.35}$$

Thus, it is possible to define the DD domain channel matrix $\mathbf{H} \in \mathbb{C}^{MN \times MN}$ as

$$\mathbf{H} = \big(\mathbf{F}_N \otimes \mathbf{G}_{rx}\big)\mathbf{G}\big(\mathbf{F}_N^H \otimes \mathbf{G}_{tx}\big). \tag{3.36}$$

Hence, the input-output relation in the DD domain can be written as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z}. \tag{3.37}$$

This relation is valid under the assumption of fine delay and Doppler resolutions. Therefore, a more general expression for the DD domain channel matrix will be provided in the following chapter in order to take into account also the effect of fractional delays and Doppler shifts. As it will be discussed this is crucial in the design of high-accuracy channel estimation algorithms.

Chapter 4

Channel estimation: the state of the art

In this chapter the channel estimation problem is formulated and some state-of-the-art algorithms are recalled. Channel estimation is a crucial aspect of wireless systems, as reliable data detection is possible only after equalization. This process requires the availability of the Channel State Information at Receiver (CSIR), thus channel estimation algorithms are needed to cater to this requirement.

4.1 Introduction

4.1.1 Problem context

As mentioned, this thesis is focused on the channel estimation problem with fractional delays and Doppler shifts and no prior information about the number of propagation paths [8, 9]. In this scenario, the transmitter sends a pilot symbol in the DD domain, and then, during a channel coherence window, the receiver processes the received pilot frame by running a channel estimation algorithm to extrapolate Channel State Information (CSI) and, once the estimated channel matrix $\hat{\mathbf{H}}$ is obtained, uses the estimate to detect the incoming data frames.

In the literature, the channel estimation problem is often considered in the *integer DDs scenario*, which is facilitated by the assumption of having a channel with delays and Doppler shifts that are multiples of delay and Doppler resolutions. In this case, there is no interference between adjacent DD bins, and a simple threshold method can be used to obtain good performances, as shown in [10]. On the other hand, in practical OTFS systems, this assumption may not hold. This motivates the study of *fractional DDs scenario*, which is more challenging, since the received replica of the pilot symbol in the DD domain spreads into adjacent bins after 2D sampling of the DD domain [1, 11]. This phenomenon is known as Inter-path Interference (IPI). It is

important to understand the effects of IPI caused by fractional DDs and to perform IPI cancellation to recover CSI with high accuracy.

4.1.2 State of the Art

Channel estimation in OTFS systems has been widely investigated in the literature since the detection performance of the OTFS receiver is based on the estimated channel matrix [12, 13, 14, 15]. In [10], a threshold method with embedded pilot for channel estimation is proposed, achieving good performances in both integer DDs and fractional Doppler scenarios. Channel estimation in massive Multiple Input Multiple Output (MIMO) has been investigated in [16] where additional sparsity due to a large number of transmit and receive antennas allows the channel estimation problem to become a simpler sparse recovery problem taking advantage of the angular domain. Low-complexity channel estimation schemes were considered in single-antenna systems in [8], where the Modified Maximum Likelihood Estimator (M-MLE) and Two-Step Estimator (TSE) algorithms were introduced. The first is based on the asymptotic orthogonality of a delay-Doppler parameter matrix for large OTFS frames ($M, N \rightarrow \infty$), which allows for the simplification of the cost function. On the other hand, the second one has lower complexity and relies on disjoint delay and Doppler estimation. Under the assumption of having good separability of the received pilot replica due to fine delay and Doppler resolutions, in both algorithms the IPI is cancelled and at each iteration a residual received vector is used for estimating channel parameters of the next path. Similarly, IPI cancellation is performed in the low-complexity channel estimation scheme for the discrete Zak Transform based OTFS (DZT-OTFS) proposed in [17]. Sparse Bayesian Learning (SBL) algorithms for channel estimation are also considered in [18, 19], but the M-MLE and TSE algorithms are shown to perform better then SBL in [8] when delay and Doppler resolutions are fine enough. In [20], different channel estimation schemes for the fractional DDs case are proposed with a focus on improving the spectrum efficiency and reducing the Peak-to-average Power Ratio (PAPR) by sending pilots in the TF domain. The problem of high-precision channel estimation in IPI scenarios has been addressed in [11], where an iterative Delay-Doppler Inter-path Interference Cancellation (DDIPIC) algorithm was proposed. In this case, the IPI cancellation is performed indirectly through refinement procedures carried out at each iteration. These refinements of channel parameters take time, increasing the computational cost of the channel estimation algorithm. However, in [11] the DDIPIC algorithm is shown to perform better than the M-MLE algorithm.

In the remainder of the thesis, for the aforementioned reasons, only the DDIPIC algorithm has been considered as a benchmark.

4.2 Problem formulation

The purpose of channel estimation is to determine channel parameters, based on transmitted pilot signals. In this section, both the pilot signal and the corresponding observation models are presented.

4.2.1 Pilot signal model

The transmitter sends a pilot symbol $x_p = \sqrt{E_p}$ in the DD domain, where E_p is the energy of the transmitted pilot signal. The corresponding pilot frame is given as

$$X[m,n] = \begin{cases} x_p & m = m_p,\ n = n_p, \\ 0 & \text{otherwise}, \end{cases} \tag{4.1}$$

where $m = m_p$, $n = n_p$ is the Delay-Doppler Resource Element (DDRE) in which the pilot is sent. The received pilot frame is obtained according to the input-output relation in the DD domain. As mentioned earlier, delay and Doppler resolutions may not be enough to approximate the true values of channel parameters with on-grid values. In this case, the channel is characterized by fractional delays and Doppler shifts that are source of inter-path interference (IPI). In order to take into account this effect a more general expression for the DD domain channel matrix, assuming rectangular shaping filters of duration T and amplitude $1/\sqrt{T}$ at both transmitter and receiver side, is given by [8, 11, 21]

$$\mathbf{H} = \sum_{i=1}^{P} g_i \mathbf{G}_i(\tau_i, \nu_i), \tag{4.2}$$

where $\mathbf{G}_i(\tau_i, \nu_i) \in \mathbb{C}^{MN \times MN}$ is a matrix that captures the effect of the propagation delay and the Doppler shift of the i-th path and for $l', l'' = 0, 1, .., M-1$, $k', k'' = 0, 1, ..., N-1$ is computed as [8, 11]

$$G_i[k'M+l', k''M+l''] = \frac{e^{-j2\pi\tau_i\nu_i}}{MN} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} r_{k'',l'}(m) e^{j2\pi\left(\frac{m}{M}\left(l'-l''-\frac{M\tau_i}{T}\right)-\frac{n}{N}\left(k'-k''-\frac{N\nu_i}{\Delta f}\right)\right)}, \tag{4.3}$$

where

$$r_{k'',l'}(m) = \sum_{s=-m}^{M-1-m} e^{j2\pi s\frac{l'}{M}} \Bigg[\left(1-\frac{\tau_i}{T}\right) e^{j\pi\left(1+\frac{\tau_i}{T}\right)\left(\frac{\nu_i}{\Delta f}-s\right)} \operatorname{sinc}\left(\left(1-\frac{\tau_i}{T}\right)\left(\frac{\nu_i}{\Delta f}-s\right)\right) + \frac{\tau_i}{T} e^{-j2\pi\frac{k''}{N}} e^{j\pi\left(\frac{\tau_i}{T}\right)\left(\frac{\nu_i}{\Delta f}-s\right)} \operatorname{sinc}\Big(\tau_i\big(\nu_i - s\Delta f\big)\Big) \Bigg]. \tag{4.4}$$

A key parameter for the design of channel estimation algorithms is Pilot Signal-to-noise Ratio (PSNR). It can be noticed that, if the channel gains are normalized such that $\sum_{i=1}^{P} |g_i|^2 = 1$, the received pilot signal is equal to that of the transmitted one. Under this hypothesis, PSNR can be easily computed considering that the pilot signal power is the total pilot energy E_p divided by the signal duration NT. Moreover, the noise power is the product of the noise power spectral density N_0 and the signal bandwidth $M\Delta f$. Thus,

$$\text{PSNR} = \frac{E_p/NT}{M\Delta f N_0} = \frac{E_p}{MNN_0}. \tag{4.5}$$

4.2.2 Observation model

In some cases, as in [11], it is better to write the input-output relation in Equation (3.37) as

$$\mathbf{y} = \mathbf{R}\big(\boldsymbol{\tau}, \boldsymbol{\nu}\big)\mathbf{g} + \mathbf{z}, \tag{4.6}$$

where $\mathbf{g} = [g_1 \ g_2 \ ... \ g_P]^T \in \mathbb{C}^{P\times 1}$ is the channel gains vector and $\mathbf{R}\big(\boldsymbol{\tau}, \boldsymbol{\nu}\big) = [\mathbf{r}_1 \ \mathbf{r}_2 \ ... \ \mathbf{r}_P] \in \mathbb{C}^{MN\times P}$ is a matrix called Costituent Delay-Doppler Parameter Matrix (CDDPM). Notice that $\boldsymbol{\tau}$ and $\boldsymbol{\nu}$ are the vectors containing propagation delays and Doppler shifts of all paths, respectively. Each column of $\mathbf{R}\big(\boldsymbol{\tau}, \boldsymbol{\nu}\big)$ is related to a different propagation path and is given by

$$\mathbf{r}_i(\tau_i, \nu_i) = \mathbf{G}_i(\tau_i, \nu_i)\mathbf{x}. \tag{4.7}$$

The main advantage of this form is that, exploiting the knowledge of the transmitted frame, channel parameters are put in evidence.

4.3 The integer DDs case

In this section the goal is to present a very simple algorithm that can be used for channel estimation in the case of integer delays and Doppler shifts. In this case we assume that delay and Doppler resolutions are fine enough such that true channel parameters are well approximated by on-grid values. The following subsection aims to describe the well-known threshold method.

4.3.1 Threshold method

The main idea is to look at the amplitude of the received DD samples. A path with delay and Doppler shifts equal to the amount of delay shift or Doppler shift of the pilot position in the DD plane is detected if the amplitude of the delay-Doppler Cell Under Test (CUT) is above a given threshold. The number of detected paths corresponds

to the number of DD cells with amplitude higher than the threshold. A reasonable value for the threshold is given by $\mathcal{T} = 3\sqrt{N_0}$. This value is chosen considering the Gaussian model for noise; therefore, a reasonable estimate of the maximum noise amplitude with 99.73% of accuracy is given by three times the standard deviation. In more details, if $|Y[m_p + l_i, n_p + k_i]| > \mathcal{T}$ a path with delay $l_i\tau_{\text{res}}$ and Doppler $k_i\nu_{\text{res}}$ is detected.

The estimate of the channel gain for the i-th path is given by

$$g_i = \frac{Y[m_p + l_i, n_p + k_i]}{x_p z^{k_i m_p}}. \tag{4.8}$$

It is clear that, if the resolution is not enough, the inter-path interference makes this algorithm find a huge number of paths, overestimating P. Thus, channel estimation algorithms that take into account fractional DDs are investigated in the literature in order to provide better performance.

4.4 Fractional DDs case

As mentioned before, several algorithms were introduced in the literature but, for the goal of this thesis, only the delay-Doppler inter-path interference cancellation algorithm can be considered. The details of this algorithm are presented in the following subsections.

4.4.1 DDIPIC algorithm

DDIPIC algorithm is an iterative tool that detects possible paths and provides estimates as the iteration progresses. This algorithm focuses on the idea of IPI cancellation through multiple refinement steps performed during the process.

Maximum likelihood estimate of channel parameters

Given the input-output relation in Equation (4.6), the Maximum Likelihood (ML) estimate of the channel parameters (gains, propagation delays and Doppler shifts) is given by

$$(\hat{\mathbf{g}}, \hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\nu}}) = \arg\min_{\mathbf{g},\boldsymbol{\tau},\boldsymbol{\nu}} \|\mathbf{y} - \mathbf{R}\big(\boldsymbol{\tau}, \boldsymbol{\nu}\big)\mathbf{g}\|^2. \tag{4.9}$$

Although the minimization problem depends on three parameters, it can be simplified performing a two-step ML estimate of the unknowns. In particular, once the propagation delays and Doppler shifts are known, the channel gains vector can be obtained by

$$\hat{\mathbf{g}} = \arg\min_{\mathbf{g}} \|\mathbf{y} - \mathbf{R}\big(\hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\nu}}\big)\mathbf{g}\|^2. \tag{4.10}$$

In order to minimize the objective function, one can compute the derivative as

$$\begin{aligned}\frac{d}{d\mathbf{g}}\|\mathbf{y}-\mathbf{R}(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{g}\|^2 &= \frac{d}{d\mathbf{g}}\Big(\mathbf{y}-\mathbf{R}(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{g}\Big)^H\Big(\mathbf{y}-\mathbf{R}(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{g}\Big)\\ &= -2\mathbf{R}^H(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\Big(\mathbf{y}-\mathbf{R}(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{g}\Big),\end{aligned} \tag{4.11}$$

and the setting it to zero. This leads to

$$\mathbf{R}^H(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{y} = \mathbf{R}^H(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{R}(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{g}. \tag{4.12}$$

Thus the channel gain vector estimate is given as

$$\hat{\mathbf{g}} = \Big(\mathbf{R}^H(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{R}(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\Big)^{-1}\mathbf{R}^H(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{y}. \tag{4.13}$$

Estimation of the delays and Doppler shifts can be done by returning to the original problem and substituting the expression for the channel gain vector. In particular, the objective function can be written as

$$\begin{aligned}\|\mathbf{y}-\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{g}\|^2 = \Big(\mathbf{y}-\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{g}\Big)^H\Big(\mathbf{y}-\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{g}\Big) =\\ \mathbf{y}^H\mathbf{y}-\mathbf{y}^H\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{g}-\Big(\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{g}\Big)^H\mathbf{y}+\Big(\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{g}\Big)^H\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{g} =\\ \mathbf{y}^H\mathbf{y}-\mathbf{y}^H\mathbf{p}_{\mathbf{R}}\mathbf{y}-\mathbf{g}^H\mathbf{R}^H(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{y}+\mathbf{g}^H\mathbf{R}^H(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{g},\end{aligned} \tag{4.14}$$

where $\mathbf{p}_{\mathbf{R}} = \mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\Big(\mathbf{R}^H(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\Big)^{-1}\mathbf{R}^H(\boldsymbol{\tau},\boldsymbol{\nu})$ is the orthogonal projector over the columns of the CDDPM matrix. It can be noticed that

$$\mathbf{g}^H = \Big(\mathbf{R}^H(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{y}\Big)^H\Big(\mathbf{R}^H(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\Big)^{-1} = \mathbf{y}^H\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\Big(\mathbf{R}^H(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\Big)^{-1}. \tag{4.15}$$

Therefore, we get

$$\|\mathbf{y}-\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu})\mathbf{g}\|^2 = \mathbf{y}^H\mathbf{y}-2\mathbf{y}^H\mathbf{p}_{\mathbf{R}}\mathbf{y}+\mathbf{y}^H\mathbf{p}_{\mathbf{R}}^2\mathbf{y}, \tag{4.16}$$

and exploiting the idempotence property of the orthogonal projector we finally obtain

$$(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}}) = \arg\min_{\boldsymbol{\tau},\boldsymbol{\nu}}\mathbf{y}^H\mathbf{y}-\mathbf{y}^H\mathbf{p}_{\mathbf{R}}\mathbf{y}. \tag{4.17}$$

This minimization problem is equivalent to the following maximization problem

$$(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}}) = \arg\max_{\boldsymbol{\tau},\boldsymbol{\nu}}\Phi\big(\mathbf{R}(\boldsymbol{\tau},\boldsymbol{\nu}),\mathbf{y}\big), \tag{4.18}$$

where, the cost function $\Phi\big(\mathbf{R}\big(\boldsymbol{\tau},\boldsymbol{\nu}\big),\mathbf{y}\big)$ is defined as

$$\Phi\big(\mathbf{R}\big(\boldsymbol{\tau},\boldsymbol{\nu}\big),\mathbf{y}\big) = \mathbf{y}^H \mathbf{p}_{\mathbf{R}} \mathbf{y}. \tag{4.19}$$

The algorithm flow

The details of the DDIPIC algorithm are provided in Algorithm 1. The algorithm proceeds iteratively for a maximum of $P_{\max}$ iterations and the CDDPM matrix is initialized as $\mathbf{R}\big(\boldsymbol{\tau},\boldsymbol{\nu}\big) = \mathbf{0}$. At each iteration, a possible path is detected and the channel parameters related to the detected path are estimated. It is possible to identify the following steps:

1. *Coarse estimation phase (line 2)*: the cost function in Equation (4.19) is maximized over the DD grid of integer multiples of delay and Doppler resolutions. In this way, a path with delay and Doppler $\tau^{(i)}_{\text{coarse}}, \nu^{(i)}_{\text{coarse}}$ is detected. During this phase, the search area is limited to $(\tau,\nu) \in \{[0, L\tau_{\text{res}}] \times [-K\nu_{\text{res}}, K\nu_{\text{res}}]\}$ where $L = \lceil \frac{\tau_{\max}}{\tau_{\text{res}}} \rceil$ and $K = \lceil \frac{\nu_{\max}}{\nu_{\text{res}}} \rceil$ are the maximum normalized delay and Doppler shift computed taking into account maximum channel parameter values $(\tau_{\max}, \nu_{\max})$.

2. *Fine estimation phase (lines 5–11)*: an iterative procedure runs for a maximum of s_{max} iterations to provide a better estimate of the fractional DDs. In each iteration, the cost function in Equation (4.19) is maximized over a search area comprising fractional delays and Dopplers around, for $s = 1$, the coarse estimate obtained in step 1 or, for $s > 1$, the fine estimate obtained in the previous iteration. The search space is narrowed down as the iteration progresses, reducing the delay and Doppler search widths (lines 6–7). The fine estimation procedure ends when the stopping criteria for τ and ν are met.

   In particular, the search space for the fine estimation procedure is referred to as $F^{(s)}_{DD}$ and we define $\Gamma = \{-\lfloor \frac{m_\tau}{2} \rfloor, ..., 0, ..., \lfloor \frac{m_\tau}{2} \rfloor\}$ and $\Lambda = \{-\lfloor \frac{n_\nu}{2} \rfloor, ..., 0, ..., \lfloor \frac{n_\nu}{2} \rfloor\}$ where m_τ and n_ν are design parameters and assume integer values. At the s-th iteration the search widths in the DD plane are $W^{(s)}_\tau = \frac{\tau_{\text{res}}}{m_\tau^{s-1}}$ and $W^{(s)}_\nu = \frac{\nu_{\text{res}}}{n_\nu^{s-1}}$. Thus, search widths are exponentially reduced from one iteration to the next one. It is worth noticing that after the Cartesian product of the two sets, only positive delays need to be considered. The fine estimation phase ends if $\left|\hat{\tau}^{(s)}_{\text{fine}} - \hat{\tau}^{(s-1)}_{\text{fine}}\right| < \epsilon_\tau$ and $\left|\hat{\nu}^{(s)}_{\text{fine}} - \hat{\nu}^{(s-1)}_{\text{fine}}\right| < \epsilon_\nu$ are satisfied. The thresholds ϵ_τ and ϵ_ν for the stopping criterion are chosen to achieve the desired accuracy compatibly with $s_{\max}$.

3. *Update channel gain vector (line 14)*: the channel gain vector is updated in order to include the new estimate.

Algorithm 1: DDIPIC

Input: $P_{\max},\ \epsilon,\ s_{\max},\ \epsilon_\tau,\ \epsilon_\nu,\ m_\tau,\ n_\nu,\ \mathbf{y}$

1 **for** $i = 1$ **to** $P_{\max}$ **do**
2 $\quad \tau^{(i)}_{\text{coarse}}, \nu^{(i)}_{\text{coarse}} = \arg\max_{(\tau,\nu)\in I_{DD}} \Phi(\mathbf{R}(\tau,\nu),\mathbf{y})$;
3 $\quad \hat{\tau}^{(0)}_{\text{fine}} = \tau^{(i)}_{\text{coarse}}$;
4 $\quad \hat{\nu}^{(0)}_{\text{fine}} = \nu^{(i)}_{\text{coarse}}$;
5 $\quad$ **for** $s = 1$ **to** $s_{\max}$ **do**
6 $\quad\quad W^{(s)}_\tau = \frac{\tau_{\text{res}}}{m^{s-1}_\tau}$: delay search width;
7 $\quad\quad W^{(s)}_\nu = \frac{\nu_{\text{res}}}{n^{s-1}_\nu}$: Doppler search width;
8 $\quad\quad F^{(s)}_{DD} = \left\{W^{(s)}_\tau \Gamma + \hat{\tau}^{(s-1)}_{\text{fine}}\right\} \times \left\{W^{(s)}_\nu \Lambda + \hat{\nu}^{(s-1)}_{\text{fine}}\right\}$;
9 $\quad\quad \hat{\tau}^{(s)}_{\text{fine}}, \hat{\nu}^{(s)}_{\text{fine}} = \arg\max_{(\tau,\nu)\in F^{(s)}_{DD}} \Phi(\mathbf{R}(\tau,\nu),\mathbf{y})$;
10 $\quad\quad$ **if** $s = s_{\max}$ ***or*** $\left(\left|\hat{\tau}^{(s)}_{fine} - \hat{\tau}^{(s-1)}_{fine}\right| < \epsilon_\tau$ ***and*** $\left|\hat{\nu}^{(s)}_{fine} - \hat{\nu}^{(s-1)}_{fine}\right| < \epsilon_\nu\right)$ **then**
11 $\quad\quad\quad$ **break**;
12 $\quad \hat{\tau}[i] = \hat{\tau}^{(s)}_{\text{fine}}$;
13 $\quad \hat{\nu}[i] = \hat{\nu}^{(s)}_{\text{fine}}$;
14 $\quad \hat{\mathbf{g}} = \left(\mathbf{R}^H(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{R}(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\right)^{-1}\mathbf{R}^H(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\mathbf{y}$;
15 $\quad \mathcal{E}^{(i)} = \mathbf{y} - \mathbf{R}(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\hat{\mathbf{g}}$;
16 $\quad$ **if** $i > 1$ **then**
17 $\quad\quad$ **if** $\|\mathcal{E}^{(i)}\| < \epsilon$ **then**
18 $\quad\quad\quad \hat{P} = i$;
19 $\quad\quad\quad$ **break**;
20 $\quad\quad$ **else**
21 $\quad\quad\quad$ Refinement procedure $(i_{\text{ref}} = 1, ..., i)$;

22 **Output:** $\hat{\mathbf{H}} = \sum_{i=1}^{\hat{P}} \hat{g}_i \mathbf{G}_i(\hat{\tau}_i, \hat{\nu}_i)$;

4. *Check stopping criterion (line 15-19)*: a residue vector is computed as $\mathcal{E}^{(i)} = \mathbf{y} - \mathbf{R}(\hat{\boldsymbol{\tau}},\hat{\boldsymbol{\nu}})\hat{\mathbf{g}}$ and if $\|\mathcal{E}^{(i)}\| < \epsilon$ is satisfied the algorithm stops and a number of $\hat{P}$ paths are detected, otherwise, before starting the search for a new path, all previous estimates are refined as described in step 5. The convergence tolerance parameter ϵ for the stopping criterion is chosen not to underestimate the number of propagation paths, not to miss those with smaller amplitudes, and not to overestimate the number of propagation paths due to residual IPI or noise. Since the noise is Gaussian, a reasonable value could be set taking into account the fact that the number of symbols per frame is MN, and the noise variance is N_0, hence $\epsilon = 3\sqrt{MNN_0}$.

5. *Refinement procedure (line 20-21)*: all coarse and fine estimates are performed again for all the previously detected paths until the current one. In particular, at the i-th iteration lines 2–13 are run again iteratively for $i_{\text{ref}} = 1, ..., i$.

6. *Generate channel matrix estimate (line 22)*: the estimate of the DD channel matrix is provided according to channel parameter estimates.

Chapter 5

Channel estimation: the proposed approach

5.1 Contributions

In this chapter a variant of the benchmark algorithm proposed in [22] is developed. The key idea is to reduce latency due to refinements of the estimates and computational complexity of the cost function in the detection phase. The main difference of the proposed algorithm with respect to the DDIPIC is that it operates in two phases:

1. *search phase*: a certain number of possible propagation paths are detected and a first estimate of channel parameters is provided,

2. *refinement phase*: the found paths are refined, looking for possible false alarms to be discarded, until the stopping criterion is met. This procedure allows for finding the correct number of paths while improving the accuracy of the estimates.

The main contributions of the proposed approach can be summarized as follows:

- *Progressive IPI cancellation*: instead of discovering possible paths and performing refinements in each iteration maximizing the observation-based cost function as in [11], it is convenient to perform maximization of a Residue-based Cost Function (RCF) at each iteration. This allows progressive cancellation of the IPI that helps providing a first channel estimate aimed to reduce the residue energy as iteration progresses achieving high accuracy in channel matrix reconstruction.

- *Global refinement of the estimates*: the main drawback of the state-of-the-art DDIPIC algorithm is that it performs refinement procedures at each iteration, as discussed in Section 5.3. On the other hand, the proposed algorithm operates

in two distinct phases, and a single global refinement is needed, reducing the overall latency.

- *Reduced-complexity search phase*: during the iterative search phase, the proposed algorithm looks for new paths by performing a RCF maximization. This allows reducing the computational cost since the computation of the RCF does not require matrix inversion and its complexity does not depend on the iteration index. As a consequence, coarse and fine estimation procedures will take less time.

- *Resilience towards high-IPI scenarios*: when the assumption of fine delay and Doppler resolutions fails, all the propagation paths are not resolvable in the DD domain. Thus, robust low-complexity channel estimation algorithms must be designed to accommodate high-IPI scenarios. The proposed approach has been shown to offer good performance in both low- and high-IPI regimes.

The details about these key points are discussed in the rest of the chapter.

5.2 Proposed P-IPIC algorithm

In this section, the proposed variant of the DDIPIC algorithm is described. The pseudocode of the proposed Progressive Inter-path Interference Cancellation (P-IPIC) algorithm is provided in Algorithm 2. The main parameters are as for the DDIPIC algorithm and were already discussed in the previous chapter.
The proposed variant features several key differences compared to the benchmark:

- *Coarse and fine estimation based on the residuals (lines 3 and 10):* the key idea is that once a path is detected and the residue vector is obtained, during the next search, a new path is discovered based on the residual pilot signal from the previous iteration. Therefore, at the i-th iteration, the algorithm searches a new path maximizing

$$\begin{aligned}\Phi\big(\mathbf{r}(\tau,\nu),\mathcal{E}^{(i-1)}\big) &= \big(\mathcal{E}^{(i-1)}\big)^H\mathbf{r}(\tau,\nu)\Big(\mathbf{r}^H(\tau,\nu)\mathbf{r}(\tau,\nu)\Big)^{-1}\mathbf{r}^H(\tau,\nu)\mathcal{E}^{(i-1)}\\ &= \frac{\big(\mathbf{r}^H(\tau,\nu)\mathcal{E}^{(i-1)}\big)^*\mathbf{r}^H(\tau,\nu)\mathcal{E}^{(i-1)}}{\|\mathbf{r}(\tau,\nu)\|^2} = \frac{|\mathbf{r}^H(\tau,\nu)\mathcal{E}^{(i-1)}|^2}{\|\mathbf{r}(\tau,\nu)\|^2}.\end{aligned} \tag{5.1}$$

  In this way, IPI is directly canceled facilitating the next search.

- *Global refinement (line 21):* it is important to note that the residue-based solution is prone to error propagation across all estimates. This may lead to residue

Algorithm 2: Proposed P-IPIC

Input: $P_{\max},\ \epsilon,\ s_{\max},\ \epsilon_\tau,\ \epsilon_\nu,\ m_\tau,\ n_\nu,\ N_0,\ \mathbf{y}$

$\mathcal{E}^{(0)} = \mathbf{y}$;
for $i = 1$ **to** $P_{\max}$ **do**
  $\tau^{(i)}_{\text{coarse}}, \nu^{(i)}_{\text{coarse}} = \arg\max_{(\tau,\nu)\in I_{DD}} \Phi\big(\mathbf{r}(\tau,\nu), \mathcal{E}^{(i-1)}\big)$;
  $\hat{\tau}^{(0)}_{\text{fine}} = \tau^{(i)}_{\text{coarse}}$;
  $\hat{\nu}^{(0)}_{\text{fine}} = \nu^{(i)}_{\text{coarse}}$;
  for $s = 1$ **to** $s_{\max}$ **do**
    $W^{(s)}_\tau = \frac{\tau_{res}}{m_\tau^{s-1}}$: delay search width;
    $W^{(s)}_\nu = \frac{\nu_{res}}{n_\nu^{s-1}}$: Doppler search width;
    $F^{(s)}_{DD} = \left\{W^{(s)}_\tau \Gamma + \hat{\tau}^{(s-1)}_{\text{fine}}\right\} \times \left\{W^{(s)}_\nu \Lambda + \hat{\nu}^{(s-1)}_{\text{fine}}\right\}$;
    $\hat{\tau}^{(s)}_{\text{fine}}, \hat{\nu}^{(s)}_{\text{fine}} = \arg\max_{(\tau,\nu)\in F^{(s)}_{DD}} \Phi\big(\mathbf{r}(\tau,\nu), \mathcal{E}^{(i-1)}\big)$;
    if $s = s_{\max}$ ***or*** $\left(\left|\hat{\tau}^{(s)}_{fine} - \hat{\tau}^{(s-1)}_{fine}\right| < \epsilon_\tau$ ***and*** $\left|\hat{\nu}^{(s)}_{fine} - \hat{\nu}^{(s-1)}_{fine}\right| < \epsilon_\nu\right)$ **then**
      break;
  $\hat{\tau}[i] = \hat{\tau}^{(s)}_{\text{fine}}$;
  $\hat{\nu}[i] = \hat{\nu}^{(s)}_{\text{fine}}$;
  $\hat{\mathbf{g}} = \left(\mathbf{R}^H\big(\hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\nu}}\big)\mathbf{R}\big(\hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\nu}}\big) + N_0\mathbf{I}\right)^{-1}\mathbf{R}^H\big(\hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\nu}}\big)\mathbf{y}$;
  $\mathcal{E}^{(i)} = \mathbf{y} - \mathbf{R}\big(\hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\nu}}\big)\hat{\mathbf{g}}$;
  if $i > 1$ **then**
    if $\|\mathcal{E}^{(i)}\| < \epsilon$ **then**
      $\hat{P} = i$;
      break;
Refinement procedure $(i_{\text{ref}} = 1, \ldots, \hat{P}) \ \rightarrow \hat{\hat{P}}$;
Output: $\hat{\mathbf{H}} = \sum_{i=1}^{\hat{\hat{P}}} \hat{g}_i \mathbf{G}_i(\hat{\tau}_i, \hat{\nu}_i)$;

vectors with some residual peaks that are interpreted as small-amplitude paths by the algorithm. In order to avoid false alarms or multidetections (when a path is detected multiple times due to uncancelled residual IPI), the P-IPIC algorithm needs a final refinement step in which all estimates are refined while the residue is monitored. If the residue satisfies the stopping criterion, all remaining unrefined estimates are declared false alarms, and therefore discarded. Specifically, during the *search phase* a number of paths equal to $\hat{P}$ is detected. After that, during the *refinement phase* the paths are refined as in the DDIPIC algorithm with the only difference in using the cost function in Equation (4.19) and the ex-

pression for the channel gain refinement (see line 15) with a regularization term. As discussed above, during the global *refinement phase* some paths may be discarded. Therefore, $\hat{\hat{P}} \leq \hat{P}$ paths are survivors and declared alarms. The number of paths detected during the *search phase* cannot be determined a priori, but it is reasonable to assume that if the IPI is not too large and the channel consists of a sufficiently high number of propagation paths, $\hat{\hat{P}} \approx \hat{P}$. Therefore, for the latency comparison we will assume that the overestimation ratio of P during the first phase is approximately one.

- *Regularization (line 15):* altough mentioned previously, another difference from the original version of the IPI cancellation algorithm is that the Ordinary Least Squares (OLS) estimate of channel gains in Equation (4.13) is replaced by a Regularized Least Squares (RLS) estimate to make matrix inversion more stable. This is due to the fact that, due to residual IPI, if some paths are detected multiple times, the CDDPM matrix may have some almost collinear columns. The RLS estimate of the channel gains vector is given by

$$\hat{\mathbf{g}} = \left(\mathbf{R}^H(\hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\nu}})\mathbf{R}(\hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\nu}}) + N_0\mathbf{I}\right)^{-1}\mathbf{R}(\hat{\boldsymbol{\tau}}, \hat{\boldsymbol{\nu}})\mathbf{y} \tag{5.2}$$

  where $\mathbf{I}$ denotes the identity matrix.

5.3 Comparison with the benchmark

5.3.1 Latency comparison

As mentioned before, the key advantage of the proposed approach is that it requires a single final refinement procedure compared to the $\hat{P} - 1$ refinements performed by the DDIPIC algorithm. However, it is worth to roughly evaluate the latency difference between the two approaches. The comparison is shown in Table 5.1, where we assumed that the P-IPIC algorithm does not overestimate the number of paths significantly during the first iterative procedure ($\hat{\hat{P}} \approx \hat{P}$). Under this hypothesis, and denoting by T_{coarse} the average time needed to perform a coarse estimate and with T_{fine} the average time needed to perform a fine estimate, we can make the following observations.

1. DDIPIC: it performs a coarse and fine estimation for every path in a time equal to $\hat{P}(T_{\text{coarse}} + T_{\text{fine}})$. Moreover, from the second iteration to the end it performs $\hat{P} - 1$ refinement procedures of the estimates. Therefore, at each refinement procedure, it performs coarse and fine estimates i.

2. P-IPIC: it performs a coarse and fine estimation for every path in a time equal to $\hat{P}(T_{\text{coarse}} + T_{\text{fine}})$ during the *search phase*. Moreover, in the *refinement phase*

performs a second coarse and fine estimate for each path.

From this discussion, we can conclude that the time difference of the latency of the two approaches is proportional to the square of the number of estimated paths.

Algorithm	Latency
DDIPIC [11]	$(T_{\text{coarse}} + T_{\text{fine}})\hat{P} + (T_{\text{coarse}} + T_{\text{fine}}) \sum_{i=2}^{\hat{P}} i = 2(T_{\text{coarse}} + T_{\text{fine}})\hat{P} + (T_{\text{coarse}} + T_{\text{fine}})\left(\frac{\hat{P}(\hat{P}-1)}{2} - 1\right)$
P-IPIC [22]	$2(T_{\text{coarse}} + T_{\text{fine}})\hat{P}$

Table 5.1: Latency comparison of the two Inter-path Interference Cancellation (IPIC) algorithms

5.3.2 Complexity comparison

During the *search phase*, the proposed P-IPIC algorithm runs coarse and fine estimation procedures of channel delays and Dopplers performing maximization of RCF, while during the *refinement phase* maximizes the regularized version of Equation (4.19). The complexity comparison of the cost function in Equation (4.19) and the RCF is shown in Table 5.2. It can be noticed that the first advantage in computing the RCF is that it doesn't require matrix inversion but only multiplications and additions, instead the computation of the cost function in Equation (4.19) does. In order to compare the two cost functions, we can make the following observations.

1. DDIPIC cost function in (4.19): during the i-th iteration the number of required operations is
   (a) $\mathbf{y}^H\mathbf{R}(\boldsymbol{\tau}, \boldsymbol{\nu})$, $\mathbf{y} \in \mathbb{C}^{MN\times 1}$, $\mathbf{R}(\boldsymbol{\tau}, \boldsymbol{\nu}) \in \mathbb{C}^{MN\times i}$ requires MNi multiplications and $(MN-1)i$ additions. Then, $\mathbf{R}^H(\boldsymbol{\tau}, \boldsymbol{\nu})\mathbf{y}$ is simply obtained applying the Hermitian operator
   (b) $\mathbf{R}^H(\boldsymbol{\tau}, \boldsymbol{\nu})\mathbf{R}(\boldsymbol{\tau}, \boldsymbol{\nu}) \in \mathbb{C}^{i\times i}$ requires MNi^2 multiplications and $(MN-1)i^2$ additions. After that, matrix inversion has a complexity in the order of $\mathcal{O}(i^3)$
   (c) $\mathbf{y}^H\mathbf{R}(\boldsymbol{\tau}, \boldsymbol{\nu})\left(\mathbf{R}^H(\boldsymbol{\tau}, \boldsymbol{\nu})\mathbf{R}(\boldsymbol{\tau}, \boldsymbol{\nu})\right)^{-1}$ requires i^2 multiplications and $i(i-1)$ additions
   (d) The matrix multiplication of the $1 \times i$ matrix obtained in the previous step by $\left(\mathbf{y}^H\mathbf{R}(\boldsymbol{\tau}, \boldsymbol{\nu})\right)^H$ requires i multiplications and $i-1$ additions
2. P-IPIC RCF: during the i-th iteration the number of required operations is

(a) $\mathbf{r}^H(\tau,\nu)\mathcal{E}^{(i-1)}$,$\mathbf{r} \in \mathbb{C}^{MN\times 1}$,$\mathcal{E}^{(i-1)} \in \mathbb{C}^{MN\times 1}$ requires MN multiplications and $MN-1$ additions. After that, an additional multiplication is needed to compute the squared module

(b) $\|\mathbf{r}(\tau,\nu)\|^2$ requires MN multiplications and $MN-1$ additions

(c) One final division is required to compute the ratio between numerator and denominator

It can be noticed that the computation of RCF is independent of the iteration index i, while the computational complexity of the cost function in Equation (4.19) increases as $\mathcal{O}(i^3)$ due to the inversion of a $i \times i$ matrix.

In conclusion, it is worth saying a few words about the search cost of the two algorithms. The proposed P-IPIC seacrh cost is identical to that of DDIPIC since the coarse and fine estimates are performed in the same way apart from the cost function. Hence, as in [11], the search cost depends on the number of points in the integer DD grid during the coarse estimate, or the number of points in the fractional search space $F_{DD}^{(s)}$ during the fine estimate. In particular, during the coarse estimation phase, both algorithms perform $(L+1)(2K+1)$ cost function maximizations, while during the fine estimation phase, they perform at most $\left(2\lfloor\frac{m_\tau}{2}\rfloor+1\right)\left(2\lfloor\frac{n_\nu}{2}\rfloor+1\right)$ maximizations.

Cost Function	Eq. (4.19)	Eq. (5.1)
Multiplications	$(MN+1)(i^2+i)$	$2MN+1$
Additions	$(MN-1)(i^2+i)+i^2-1$	$2(MN-1)$
Total number of operations	$2MN(i^2+i)+i^2-1+\mathcal{O}(i^3)$	$4MN$

Table 5.2: Comparison of computational complexity of cost functions

5.4 Simulation results

In this section, the proposed method is evaluated and compared with the baseline DDIPIC method.

5.4.1 Simulation parameters

To investigate the performance of the proposed approach, we consider OTFS modulation with $M = 16$ and $N = 16$ with subcarrier spacing of $\Delta f = 25$ kHz and a carrier frequency $f_c = 5.1$ GHz. Thus, delay and Doppler resolutions are $\tau_{\text{res}} = 2.5$ μs, and $\nu_{\text{res}} = 1.5625$ kHz. As in [11], the parameters for running channel estimation algorithms are $P_{\max} = 15$, $s_{\max} = 10$, $\epsilon_\tau = 10^{-10}$, $\epsilon_\nu = 10^{-2}$, $m_\tau = 10$, $n_\nu = 10$. Finally, as discussed above, the convergence tolerance parameter for the stopping criterion is set according to the noise variance and OTFS frame parameters.

5.4.2 Simulation scenarios

The high mobility multipath channel is assumed to have $P = 4$ Rayleigh faded paths with exponential PDP with decaying time constant of 10 μs and a maximum delay of 10 μs. We consider two different case studies corresponding to different scenarios:

1. *Low-IPI Regime*: propagation delays and Doppler shifts assume fractional values but the paths are far enough to guarantee low IPI. Thus, all paths are separable in the DD domain.
   The delays and Doppler shifts of the paths are $\boldsymbol{\tau} = [0 \;\; 0.86\tau_{\text{res}} \;\; 2.11\tau_{\text{res}} \;\; 3.2\tau_{\text{res}}]$, $\boldsymbol{\nu} = [-1.87\nu_{\text{res}} \;\; 2.24\nu_{\text{res}} \;\; -3\nu_{\text{res}} \;\; 1.3\nu_{\text{res}}]$, respectively.

2. *High-IPI Regime*: propagation delays and Doppler shifts assume fractional values, and the paths are close enough to create high IPI.
   The propagation delays of the paths are $\boldsymbol{\tau} = [0 \;\; 2.3 \;\; 3.15 \;\; 7.7]$ μs. The maximum User Equipment (UE) speed is $v_{UE} = 190 \; \frac{\text{m}}{\text{s}} (684 \frac{\text{km}}{\text{h}})$ resulting into a maximum Doppler spread of $\nu_{\max} = \frac{v_{UE}}{c} f_c = 3.23$ kHz, where $c = 3 \cdot 10^8 \; \frac{\text{m}}{\text{s}}$ is the speed of light.
   The Doppler shifts of all paths are generated assuming Jakes' DPS using $\nu_i = \nu_{\max} cos(\theta_i)$ where $\theta_i \sim \mathcal{U}[0, 2\pi]$.

An example of a pilot frame received in both regimes is provided in Figure 5.1.

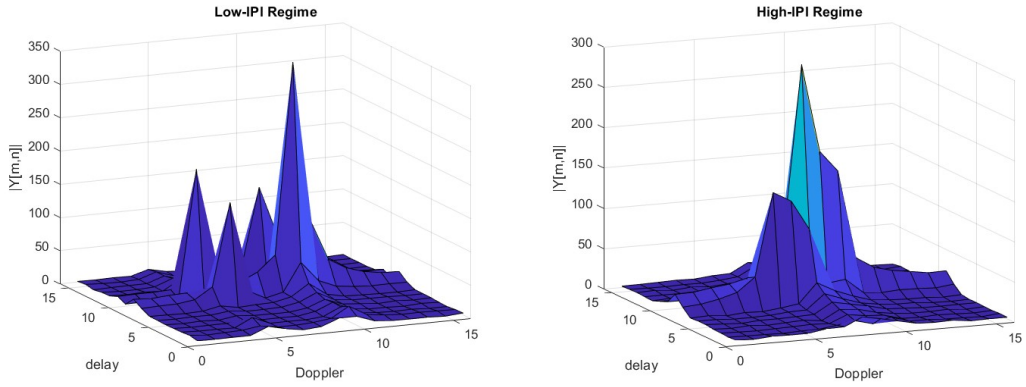

Figure 5.1: Received DD domain pilot signal.

5.4.3 Performance metrics

We will consider the following performance metrics:

- *Channel estimation:* The accuracy of the channel estimate is measured through

the Normalized Mean Square Error (NMSE) computed as

$$\text{NMSE} = \mathbb{E}\left[\frac{\|\mathbf{H} - \hat{\mathbf{H}}\|_F^2}{\|\mathbf{H}\|_F^2}\right]. \tag{5.3}$$

- *Channel parameter estimation:* The accuracy of the channel parameter estimates is also considered evaluating the Normalized Root Mean Square Error (NRMSE) as

$$\text{NRMSE}(\vartheta) = \sqrt{\mathbb{E}\left[\frac{\sum_{i=1}^{P} |\vartheta_i - \hat{\vartheta}_i|^2}{\sum_{i=1}^{P} |\vartheta_i|^2}\right]}, \tag{5.4}$$

  where ϑ is a variable that represents delay, Doppler or channel gain.

- *Model order performance:* The estimation of P will be evaluated by the average number of estimated paths and Probability Mass Function (PMF) of the number of detected paths.

- *Communication:* Communication performance of the proposed approach is evaluated by measuring the Bit Error Rate (BER) as a function of the Signal-to-noise Ratio (SNR)/bit in order to investigate the detection robustness when real channel estimation with the proposed variant is performed at the receiver. The SNR/bit is given by E_b/N_0 where E_b is the average energy per bit. We consider a QAM with $Q = 16$ constellation points and $E_b = \frac{2(Q-1)}{3\log_2 Q}$. The input-output relation for data frames is the one in Equation (3.37). At the receiver, a Linear Minimum Mean Square Error (LMMSE) equalizer provides the estimated symbols as

$$\hat{\mathbf{x}} = \left(\hat{\mathbf{H}}^H\hat{\mathbf{H}} + \frac{1}{\text{SNR}}\mathbf{I}\right)^{-1}\hat{\mathbf{H}}^H\mathbf{y}, \tag{5.5}$$

  where the communication SNR is given by the ratio between the signal power and the noise power. Given $E_s = E_b \log_2 Q$ the average energy per symbol, the signal power is given by the total average energy divided by the frame duration, thus $\frac{MNE_s}{NT}$. Moreover, the average noise power is given by the product of the bandwidth and the monolateral PSD, thus $M\Delta f N_0$. Therefore, we get $\text{SNR} = \frac{MNE_s/NT}{M\Delta f N_0} = \frac{E_s}{N_0}$.

  After equalization, a symbol-by-symbol ML detector takes the decisions.

5.4.4 Results and discussion

Channel estimation performance

Figure 5.2 shows the NMSE as a function of the pilot SNR in both low- and high-IPI regimes. In the Low-IPI case at low PSNR the performance of the proposed variant and the DDIPIC algorithm are comparable, while at higher values of PSNR the proposed variant outperforms the DDIPIC algorithm, achieving much lower NMSE. On the other hand, in the high-IPI regime, the P-IPIC algorithm achieves a NMSE 1-2 dB lower than the DDIPIC, except for the case of PSNR $= 35$ dB where the NMSE performances become comparable. We attribute this performance gain to the fact that during the *search phase* possible paths are detected with the goal of reducing the residue energy, and thus to achieve high accuracy in channel matrix estimate. After that, the good channel estimate obtained in the *search phase* is used to refine the estimates and declare false alarms.

Performance of the threshold method in [10] are also presented in both cases as a baseline for comparison and it can be noticed that they are drastically degraded due to the presence of IPI and the integer DDs assumption.

Channel parameter estimation performance

Figure 5.3 shows the NRMSE of channel gains, delays and Doppler shift as a function of PSNR in a channel randomly generated according to channel models 1 and 2.

It can be noticed that in both low- and high-IPI regimes, the P-IPIC is capable of estimating channel parameters with higher accuracy. In particular, in the low-IPI regime the performance gain increases as the PSNR becomes higher. On the other side, in the High-IPI case, at low PSNR the NRMSE is high due to the fact that small amplitude paths are missed. While, at high PSNR the NRMSE for P-IPIC increases because small errors in parameter estimates during the *search phase* lead to a higher residual IPI that is interpreted as small-amplitude paths. This phenomenon reduces the overall accuracy of the estimates leading to lower performance gain. However, P-IPIC still outperforms the DDIPIC algorithm for high PSNR values.

Number of detected paths

Figure 5.4 shows the average number of detected paths as a function of PSNR. In the Low-IPI case, performances of both algorithms are comparable. At low PSNR, the average number of detected paths is below the true value ($P = 4$) because small-amplitude paths are missed. At higher PSNR values, both algorithms are able to detect the right number of paths exhibiting good detection capabilities. On the other hand, in the High-IPI case detection capabilities are still comparable but at high PSNR values

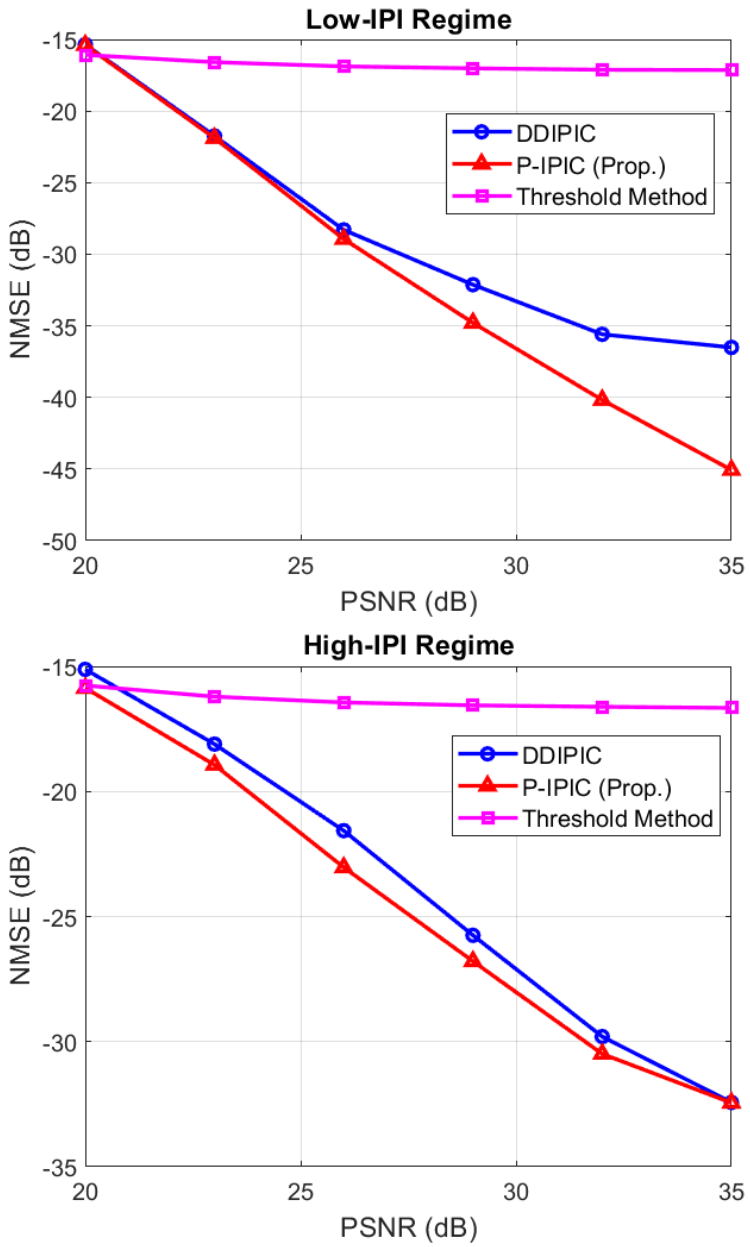

Figure 5.2: NMSE as a function of pilot SNR.

both algorithms overestimate a bit the number of propagation paths due to high residual interference effects.

Figure 5.5 shows the probability distribution among all possible values for P in the case of PSNR $= 35$ dB. In the low-IPI regime, both algorithms detect the right number of paths with a probability close to ~ 0.98. Thus, the probability of miss detection and the probability of false alarm are close to zero (probability of miss detection < 0.02).

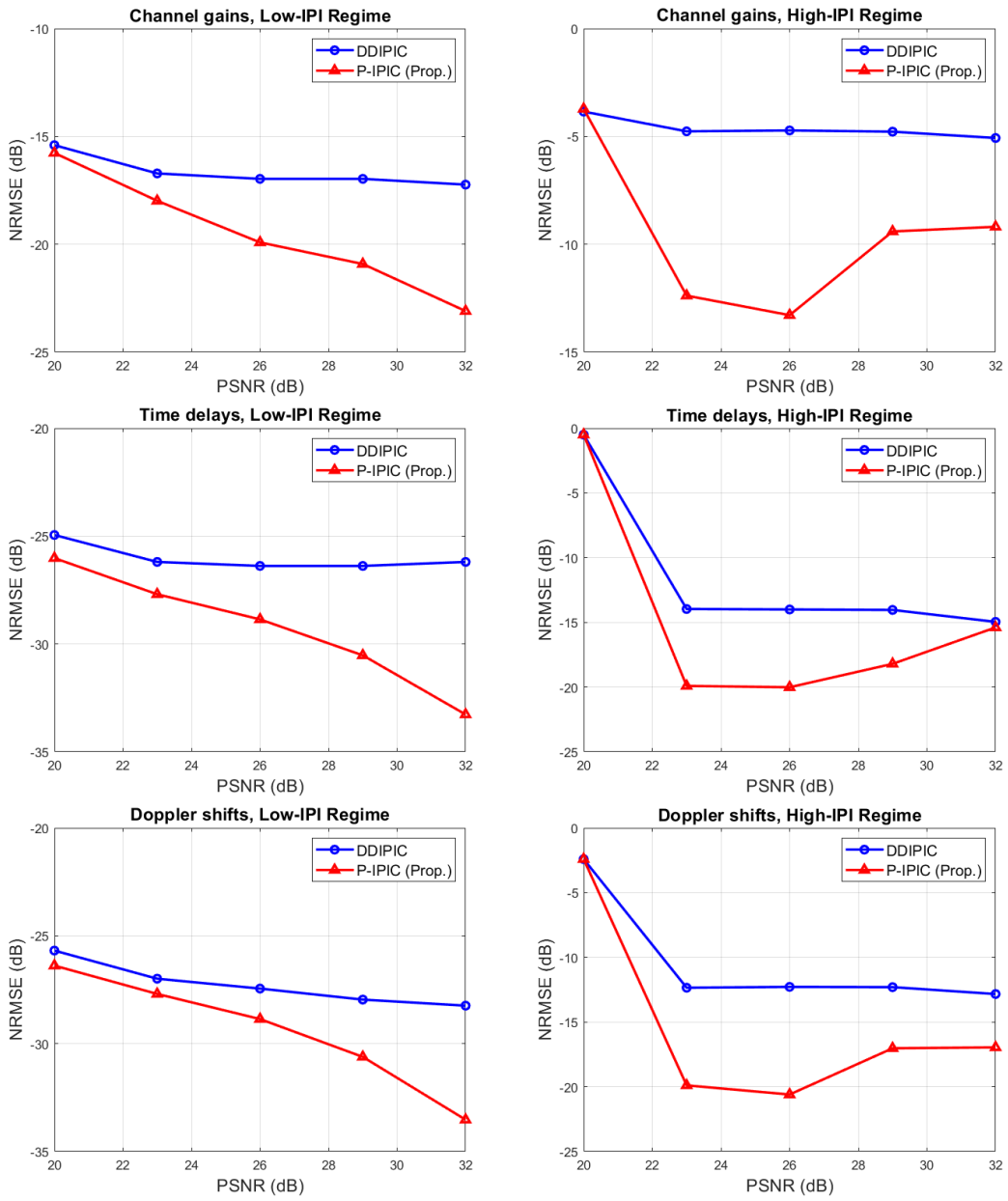

Figure 5.3: NRMSE of channel parameters as a function of pilot SNR.

In the high-IPI regime, the probability that the algorithm finds the correct number of paths is reduced to ~ 0.73 for DDIPIC and ~ 0.65 for the P-IPIC. The probability of miss detection is close to zero in both cases (~ 0.015 for the P-IPIC). In the end, the probability of false alarm is ~ 0.27 for DDIPIC and ~ 0.34 for P-IPIC.

It is possible to conclude that both algorithms exhibit good detection capabilities in extracting the correct number of paths with reasonable accuracy.

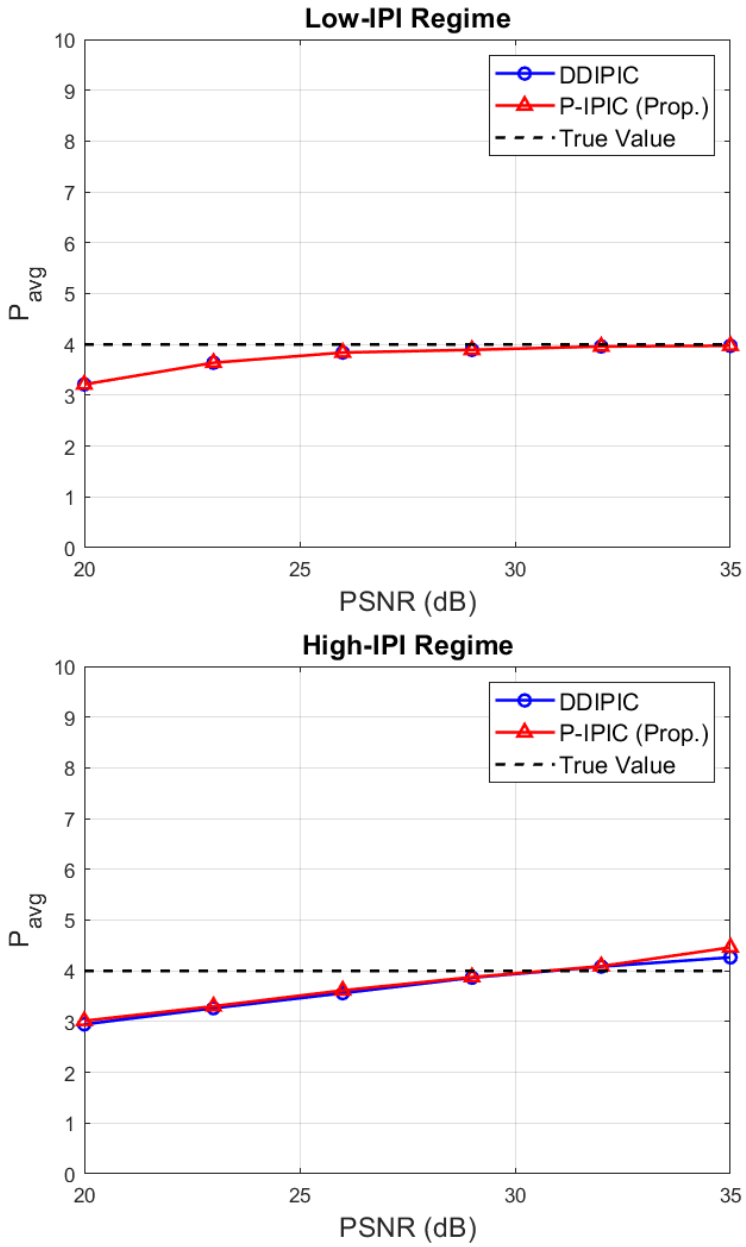

Figure 5.4: Average number of estimated paths as a function of pilot SNR.

Communication performance

BER performance with the estimated channel (PSNR = 30 dB) and with perfect channel state information at the receiver (CSIR) is provided in Figure 5.5, showing the BER performance as a function of the SNR/bit in channels corresponding to low- and high-IPI regimes. As expected, the proposed P-IPIC achieves lower bit error rates in both regimes. In addition, BER curves are closer to the ideal curves obtained with perfect

channel state information at the receiver.

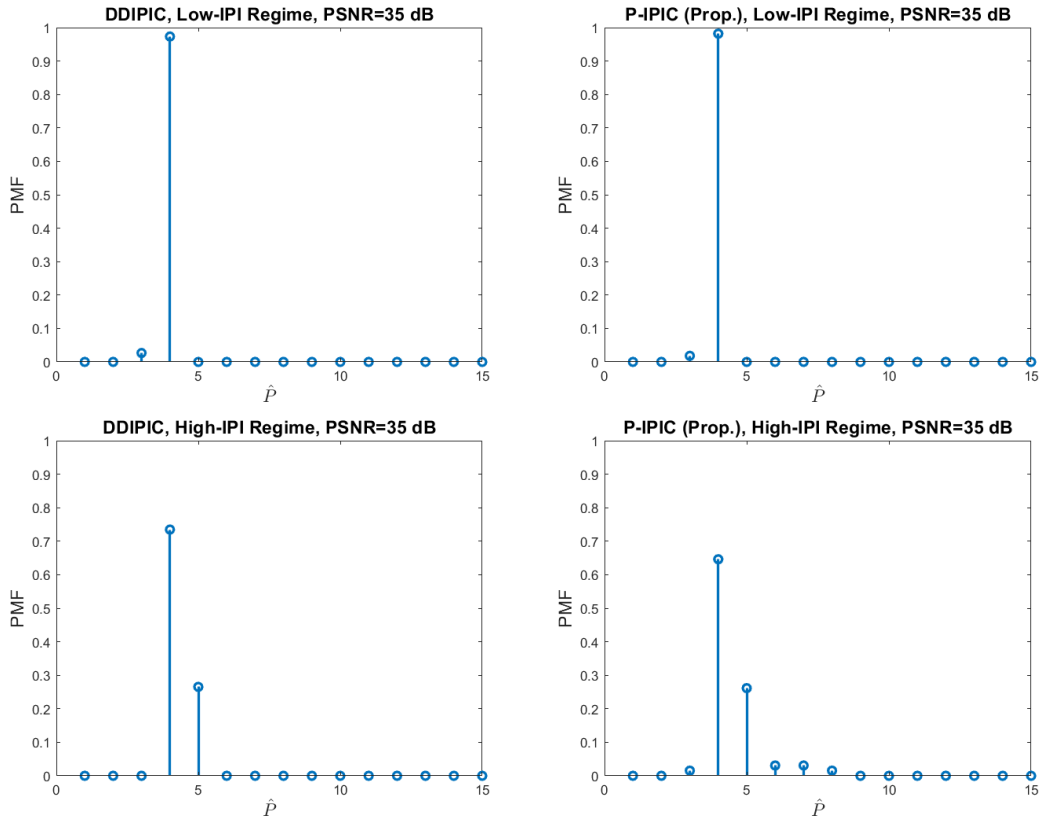

Figure 5.5: Probability mass function of the number of detected paths.

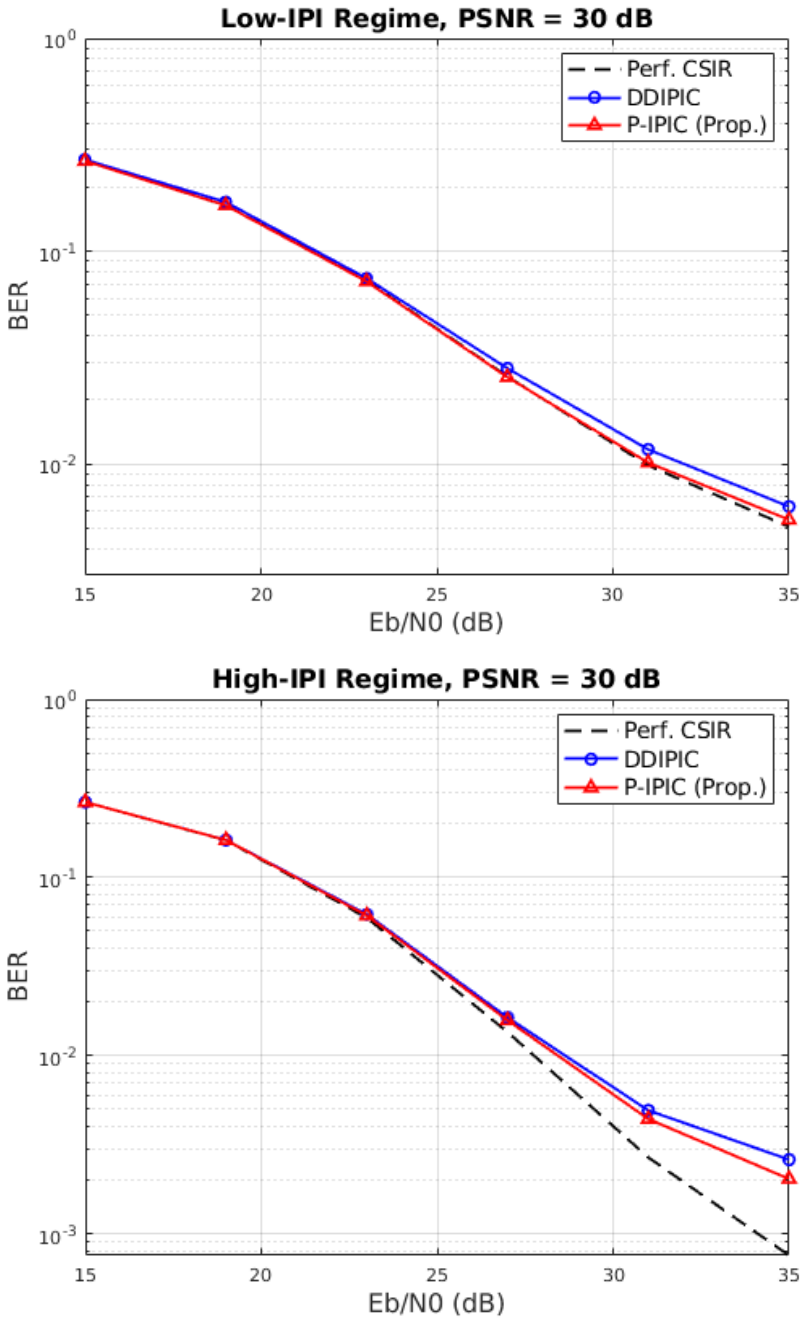

Figure 5.6: Bit error rate as a function of SNR/bit.

Chapter 6

Deep learning based approach

6.1 Why deep learning?

Deep Learning (DL) has become a revolutionary tool for advancing wireless communications, offering solutions to the complex challenges posed by emerging technologies like 6G, Internet of Things (IoT), and virtual reality. Traditionally, wireless communications have relied on model-based designs that depend heavily on accurate mathematical models. However, with the increasing complexity of modern communication systems, such as large-scale networks and unknown channel models, these traditional approaches struggle to meet the requirements of speed, efficiency, and adaptability.

Deep learning offers a powerful alternative. Unlike model-based techniques, DL excels in scenarios where accurate models are difficult to obtain and it is capable of learning complex relationships between variables without relying on predefined mathematical models, enabling systems to perform better even under uncertain conditions. This adaptability is crucial for modern wireless applications like massive MIMO and dense IoT networks, where the complexity of the systems makes traditional methods inefficient.

There are two main methodologies for integrating DL into wireless communication [23]: DL-based architecture design and DL-based algorithm design. DL-based architecture design leverages neural networks to reimagine communication system structures, moving away from the traditional block-based models that separate functions like demodulation, channel estimation, and detection. For instance, DL can optimize the entire communication process end-to-end, resulting in better performance in terms of error rates and system efficiency. An example of this is the use of DL to jointly optimize transmitter and receiver operations, breaking down the barriers between individual system components.

DL-based algorithm design focuses on improving the computational aspects of communication, such as reducing latency in large-scale networks. It enhances existing algorithms by incorporating DL's ability to speed up processes without sacrificing

performance. For example, DL-based solutions for resource allocation and channel estimation can outperform traditional algorithms, offering faster convergence and reduced computational complexity.

By breaking free from traditional model-based constraints and offering scalable, efficient solutions, DL is paving the way for more intelligent and adaptive wireless networks, particularly in the era of 6G.

In the following sections, the basics of DL are presented and the proposed DL-based solution is discussed.

6.2 Neural networks

In this section Artificial Neural Network (ANN) are introduced and classic architectures, algorithms for training and optimization are discussed.

6.2.1 Model of a neuron

A neuron is an information-processing unit that constitutes the basic building block of neural networks [24]. The artificial neuron model is shown in Figure 6.1 where it is possible to identify the following parts: a linear combiner that receives a certain number of inputs (stimulus) and computes a biased weighted sum of those inputs and an activation function that limits the output range of the neuron. Each branch with a weight is referred to as synapses and the weight measures the strength of the link. Denoting by x_j the inputs, w_{kj} the weights and b_k the bias, the output of the linear combiner for the neuron k is given as

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k, \tag{6.1}$$

where m is the number of synapses. Hence, the output of the neuron is

$$y_k = \varphi(v_k), \tag{6.2}$$

where φ denotes the activation function. There exist different type of activation functions that model the output of the neuron as a function of the local field v. Some examples are

- *Heaviside*: this activation function is also called threshold function and is given as

$$\varphi(v) = \begin{cases} 1 & v \geq 0, \\ 0 & v < 0, \end{cases}, \tag{6.3}$$

If also negative values are desired at the output one can use the signum function instead.

- *linear*: the output of the neuron is equal to the linear combiner's output. Thus,

$$\varphi(v) = v \tag{6.4}$$

It is also known as 'no activation' or 'identity function' and can be used when there is the need to span the set of real numbers at the output of the neuron.

- *Rectified Linear Unit (ReLU)*: compared to the linear activation function, the ReLU activates the neuron only if the input is positive. Thus,

$$\varphi(v) = \max(0, v) \tag{6.5}$$

In some cases, this is preferred to the linear activation function since it accelerates convergence.

- *sigmoid*: its graph is S-shaped and it is very commonly adopted. A very well-known expression is given by the logistic function

$$\varphi(v) = \frac{1}{1 + e^{-av}}, \tag{6.6}$$

where a is the slope parameter.

Figure 6.2 shows the sigmoid activation function for different slope parameter values.

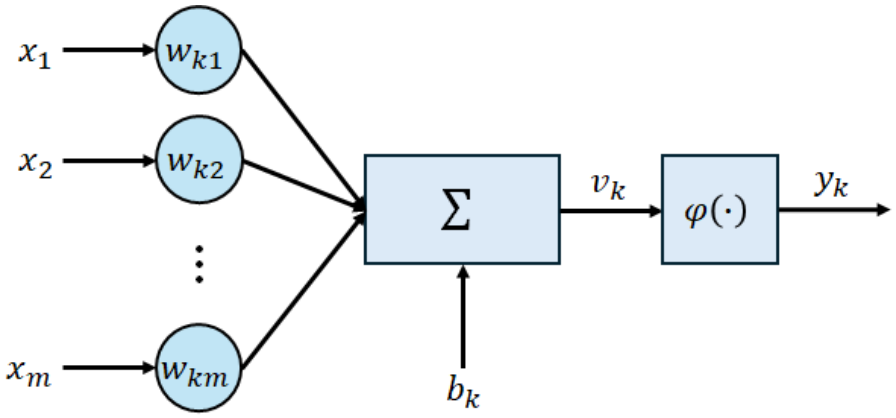

Figure 6.1: The model of an artificial neuron.

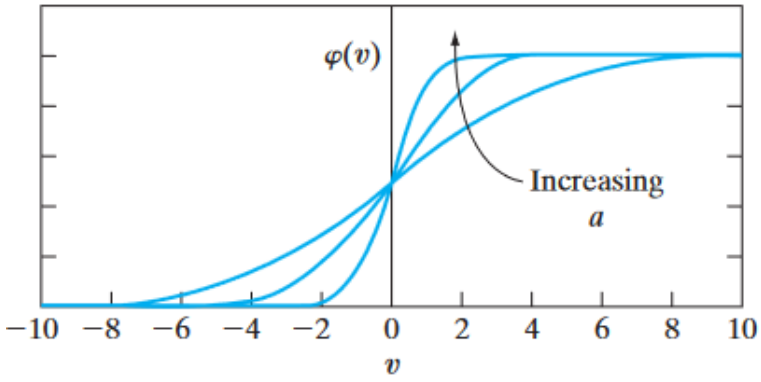

Figure 6.2: The sigmoid activation function for different values of a [24].

6.2.2 Multilayer networks

In the previous section, the model of a neuron was discussed. An artificial neural network is made of several neurons properly interconnected. A generic network is composed by:

- an *input layer* which consists of a certain number of source nodes corresponding to the network inputs,
- some *hidden layers* of neurons that are not directly visible from input and output,
- an *output layer* which is made of output nodes corresponding to network outputs.

The main role of hidden layers is to perform feature extraction and process input data in a space where input features are well separated. Figure 6.3 shows the architecture of a multilayer neural network.

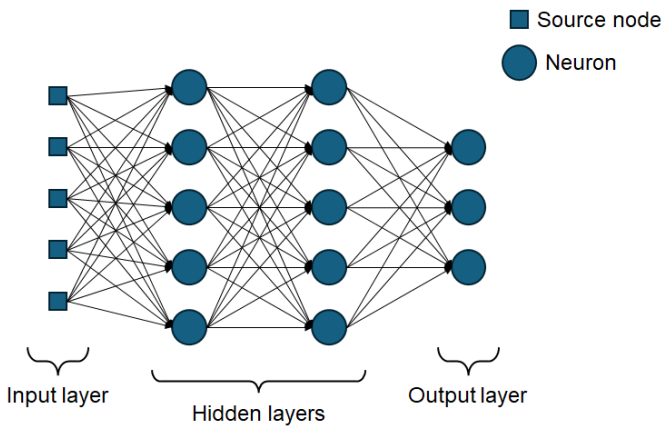

Figure 6.3: The architecture of a multilayer neural network.

6.2.3 Learning process

In order to use neural networks to perform a specific task, the network has to be submitted to a learning process. During this phase, the network weights and biases are updated in order to best fit the problem. There exist different type of learning process but it is possible to identify two distinct classes:

- *Learning with a teacher*: it is also referred to as *supervised learning*. In this case there exist a teacher that is assumed to have knowledge of input-output examples to provide to the network. The training process works as follows: an input is submitted to the network that generates an output and, after that, the teacher computes the error between the desired response and the actual response of the network. This error is used by the network to update its parameters and learn to emulate the teacher.

- *Learning without a teacher*: in this case there is no teacher to oversee the learning process. It is possible to distinguish two different approaches: reinforcement learning and unsupervised learning. In reinforcement learning, the interaction between the agent and the environment sets the learning process. In particular, the quality of an action is determined by a numerical value of 'reward', which aims to encourage the correct behaviors of the agent. On the other hand, in unsupervised learning, an algorithm analyzes unlabeled data to identify patterns or structures without external supervision. It's used for tasks like clustering similar data points or reducing dimensionality of the input.

In the following, supervised learning will be considered for the desired task.

Batch learning

In supervised learning a set of samples, referred to as batch, is considered for training. In some cases it is preferable to divide the batch in smaller mini-batches. A batch or mini-batch is passed to the network and errors are computed. Only after that, network parameters are updated. Hence, in the batch method of supervised learning, adjustments to the synaptic weights of the neural network are made after presentation of all the N_s examples in the training sample $\mathcal{T}$ that constitute one epoch of training. At the end of one epoch, training samples are randomly shuffled and a new epoch starts.

The training sample is given as

$$\mathcal{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N_s}, \tag{6.7}$$

where $\mathbf{x}(n)$ is the set of stimuli applied to the input layer and $\mathbf{d}(n)$ is the set of desired outputs.

The back-propagation algorithm

In this section the well-known back-propagation algorithm for supervised learning is introduced and the back-propagation formula is derived. The induced local field at time n at the input of the activation function associated with neuron j is

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n), \tag{6.8}$$

where m is the number of inputs and the weight $w_{j0} = b_j$ has input $y_0 = 1$. The output of the neuron is therefore

$$y_j(n) = \varphi_j(v_j(n)). \tag{6.9}$$

The back-propagation algorithm applies a correction to the synaptic weights proportional to the partial derivative $\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)}$ where $\mathscr{E}(n)$ is the cost function. The cost function can be chosen arbitrarily. For convenience, the discussion on the back-propagation algorithm will be carried on considering the total instantaneous error energy of the network as cost function [24]. In particular, the error signal produced at the output of the neuron j is $e_j(n) = d_j(n) - y_j(n)$ where $d_j(n)$ is the desired response. Thus, the instantaneous error energy of neuron j is $\mathscr{E}_j(n) = \frac{1}{2} e_j^2(n)$. Thus, the total instantaneous error energy is

$$\mathscr{E}(n) = \sum_{k \in C} \mathscr{E}_k(n), \tag{6.10}$$

where C is the set of neurons in the output layer. Applying the chain rule, the partial derivative becomes

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathscr{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \tag{6.11}$$

It can be noticed that

$$\frac{\partial \mathscr{E}(n)}{\partial e_j(n)} = e_j(n), \tag{6.12}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1, \tag{6.13}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)), \tag{6.14}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n). \tag{6.15}$$

Substituting we get

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = -e_j(n) \varphi_j'(v_j(n)) y_i(n). \tag{6.16}$$

The correction term applied to to the network weights can be written, according to the gradient descent *delta rule*, as

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = \eta \delta_j(n) y_i(n), \tag{6.17}$$

where $\delta_j(n)$ is called local gradient defined as $\delta_j(n) = -\frac{\partial \mathscr{E}(n)}{\partial v_j(n)} = -\frac{\partial \mathscr{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n))$ and η is the learning rate factor.

It is clear that in order to compute the correcting term, the error signal must be computed. In order to do this, we can make the following observations

- *Output node case*: when the j-th neuron is located in the output layer the desired response is known. Thus, the error computation is straightforward.
- *Hidden node case*: when the j-th neuron is located in a hidden layer, instead, there is no specified desired response for that layer. Thus, the desired response must be determined recursively considering all the neurons to which that hidden neuron is connected. The local gradient can be written also as

$$\delta_j(n) = -\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} \varphi_j'(v_j(n)). \tag{6.18}$$

  In order to compute the remaining partial derivative we can consider Equation (6.10) and differentiating we obtain

$$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = \sum_{k \in C} e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} = \sum_{k \in C} e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}. \tag{6.19}$$

  Since $e_k(n) = d_k(n) - \varphi_k(v_k(n))$ we get

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)). \tag{6.20}$$

  Moreover, since $v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n)$, we get

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n). \tag{6.21}$$

  Putting everything together we obtain

$$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = -\sum_{k \in C} e_k(n) \varphi_k'(v_k(n)) w_{kj}(n) = -\sum_{k \in C} \delta_k(n) w_{kj}(n). \tag{6.22}$$

  Substituting this partial derivative into the original expression we get the back-

propagation formula for the local gradient

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_{k \in C} \delta_k(n) w_{kj}(n). \tag{6.23}$$

It should be noted that the local gradient associated with the hidden neuron j depends on its activation function and the local gradients of the next-layer neurons connected to it, together with the weight. In conclusion, once the local gradient is known, the weights can be updated according to Equation (6.17).

So far, it can be noticed that the backpropagation algorithm requires two distinct steps

1. *Forward pass*: the synaptic weights remain unchanged and the output of each neuron is determined,

2. *Backward pass*: it starts at the output layer by passing backward the error signal computing recursively the local gradient of each neuron according to the backpropagation formula and then updating its weights.

In conclusion, it is important to note that the backpropagation algorithm works under the implicit assumption of having an activation function which is differentiable.

Adam optimizer

The back-propagation algorithm is ruled by the following update equation

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n), \tag{6.24}$$

where the correction term is given by the gradient descent, as discussed in the previous section.

In order to control and stabilize the convergence of the gradient descent, some optimization algorithms can be employed. A well-known optimization algorithm is the so called Adam optimizer, which means adaptive moment estimation. This algorithm combines two techniques

- *Momentum*: it is used to track gradient directions and dampen oscillations

- *Adaptive learning-rate parameter*: the learning-rate factor is adaptively controlled according to the gradient history.

The first momentum is a moving average of the gradient. In particular it is defined as

$$m_{ji}(n) = \beta_1 m_{ji}(n-1) + (1-\beta_1) \frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)}, \tag{6.25}$$

where β_1 is am hyper-parameter that controls the weight of the past gradients in the average. A typical value is $\beta_1 = 0.9$.

The Adam optimizer tracks also the variance of the gradient to adapt the learning rate. Thus, the second momentum is defined as

$$V_{ji}(n) = \beta_2 V_{ji}(n-1) + (1-\beta_2)\left(\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)}\right)^2, \tag{6.26}$$

where β_2 is an hyper-parameter controlling the decay of the gradient in the second momentum estimate. A typical value is $\beta_2 = 0.999$.

However, during first iterations, since the first and second momentums usually are initialized at zero, they tend to assume very small values. Therefore, a bias correction term is applied in order to overcome this issue. Thus, we get

$$m'_{ji}(n) = \frac{m_{ji}(n)}{1-\beta_1^n}, \tag{6.27}$$

$$V'_{ji}(n) = \frac{V_{ji}(n)}{1-\beta_1^n}. \tag{6.28}$$

Once the estimate of first and second momentum is obtained, the weight update equation can be written as

$$w_{ji}(n+1) = w_{ji}(n) - \frac{\eta}{\sqrt{V_{ji}(n)} + \epsilon} m_{ji}(n), \tag{6.29}$$

where ϵ is a small positive number used as a regularization term. A typical value may be $\epsilon = 10^{-8}$.

6.3 Application to OTFS channel estimation

In the OTFS scope, DL approaches are used for different tasks such as Inphase Quadrature Imbalance (IQI) compensation [25], data detection [26] and also channel estimation. DL is a useful tool for channel estimation, as it can be used to reduce computational cost and/or latency and to enhance the accuracy of channel estimation algorithms. In the literature different DL-based approaches are proposed. In [27] a Residual Network is used to refine the channel matrix estimated using the Orthogonal Matching Pursuit (OMP) algorithm assuming integer DDs. In [28], a RNN is used to extrapolate CSI from the interleaved pilot scheme. Other DL-based approaches are shown in [29, 30] where two different solutions are proposed to cope with different noise sources increasing the robustness of channel estimation. In [31] a DL architecture is used to directly learn the channel matrix starting from the received pilot frame.

6.3.1 FNN-based architecture

In this section the DL-based architecture is presented. As proposed in [32], the computation cost required for the computation of the cost function in Equation (5.1) is dominated by the complexity of the computation of the CDDPM column. Thus, a direct computation of the CDDPM column can be substituted by learning the relation in Equation (4.7). In this case, the idea is to use a neural network trained to learn that relation. Moreover, since the unvectorized CDDPM column has a high dynamic range in the delay-Doppler domain, it is more convenient to perform learning in the TF domain [32]. Figure 6.4 shows the comparison of the dynamic ranges of DD and TF versions of the unvectorized CDDPM column.

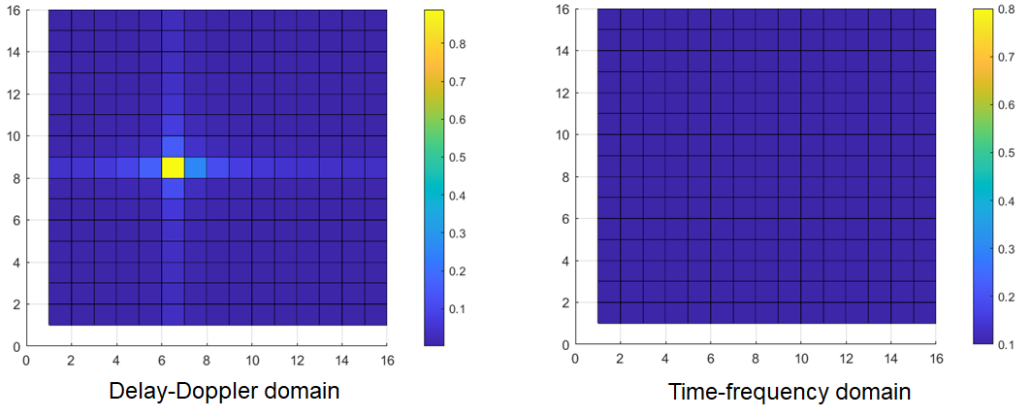

Figure 6.4: Comparison of dynamic range of DD and TF unvectorized CDDPM column.

Thus, according to [32], the DL-based architecture used to learn a column of the CDDPM is shown in Figure 6.5. The architecture is made up of two Feedforward Neural Network (FNN) used to learn the real and imaginary parts of the TF CDDPM column, respectively. In particular, the FNN is characterized by

- *Two inputs*: the network receives as input a normalized DD pair. Normalization is performed in order to have a range of values between $[0, 1]$ for the delay and $[-1, 1]$ for the Doppler shift
- *Concatenation layer*: it is used to concatenate the two inputs
- Fully connected layers: a certain number of fully connected layers is used. A layer is fully connected if all neurons in the previous layer are connected to each neuron in that layer. The activation function is ReLU a part of the output layer that adopts a linear activation function.

- *Output layer*: it provides the final output of the network. Its dimension equals MN since the TF CDDPM column has to be provided as output.

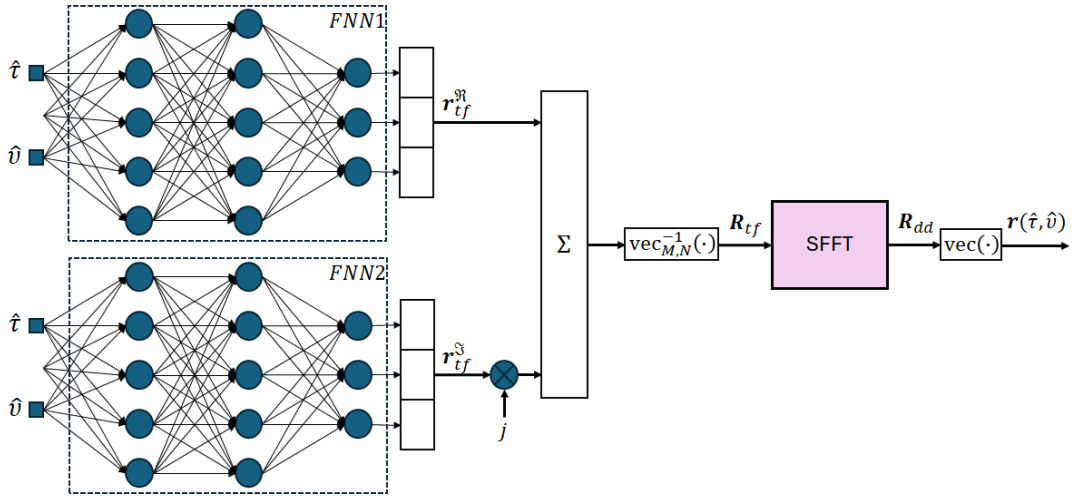

Figure 6.5: The architecture of the TF learning approach.

As depicted in Figure 6.5, FNN1 learns the real part of the TF CDDPM column, denoted as $\mathbf{r}_{tf}^{\Re} = \Re\{\mathbf{r}_{tf}\}$, while FNN2 learns the imaginary part $\mathbf{r}_{tf}^{\Im} = \Im\{\mathbf{r}_{tf}\}$. Where the TF CDDPM column is given by

$$\mathbf{r}_{tf} = \mathrm{vec}\big(\mathbf{R}_{tf}\big) = \mathrm{vec}\big(\mathbf{F}_M \mathbf{R}_{dd} \mathbf{F}_N^H\big), \tag{6.30}$$

where $\mathbf{R}_{dd} = \mathrm{vec}_{M,N}^{-1}\big(\mathbf{r}\big(\hat{\tau},\hat{\nu}\big)\big)$. Thus, the two network outputs are combined and reshaped to obtain $\mathbf{R}_{tf}$. Then, SFFT is applied to obtain the DD matrix that, once vectorized, provides the desired CDDPM column.

The number of layers and the number of neurons per layer are mentioned in Table (6.1). The input dimension of the network is 2, the number of neurons per layer is doubled at each layer. The last layer with MN neurons is used to obtain the desired output dimension.

Number of hidden layers	10
Number of neurons per hidden layer	$2^{L+1}, L =$ layer index
Input dimension	2
Output dimension	MN

Table 6.1: FNN architecture parameters.

6.3.2 Training phase

The FNN-based architecture is trained to learn the desired relation. Training is performed using the backpropagation algorithm and Adam optimizer discussed in the previous sections. The hyperparameters used during the training phase are reported in Table (6.2). The learning rate is halved every 10 epochs. The loss function is the L1 loss function. Denoting $y_k(n)$ the output of the k-the neuron in the output layer and $d_k(n)$ the desired value for that output, the L1 loss function is defined as

$$\mathscr{E}(n) = \sum_{k=1}^{MN} |d_k(n) - y_k(n)|. \tag{6.31}$$

It is worth to notice that the back-propagation formula is still valid and can be used once the local gradients of the output layer neurons are determined.

Number of training samples	200000
Mini batch size	1000
Number of epochs	100
Initial learning rate	0.001
Learning rate drop factor	0.5
Learning rate drop period	10

Table 6.2: Hyper-parameters for training.

Training is carried out offline as described above and in the same way for the two networks. Figure 6.6 shows the training loss as a function of the iteration index.

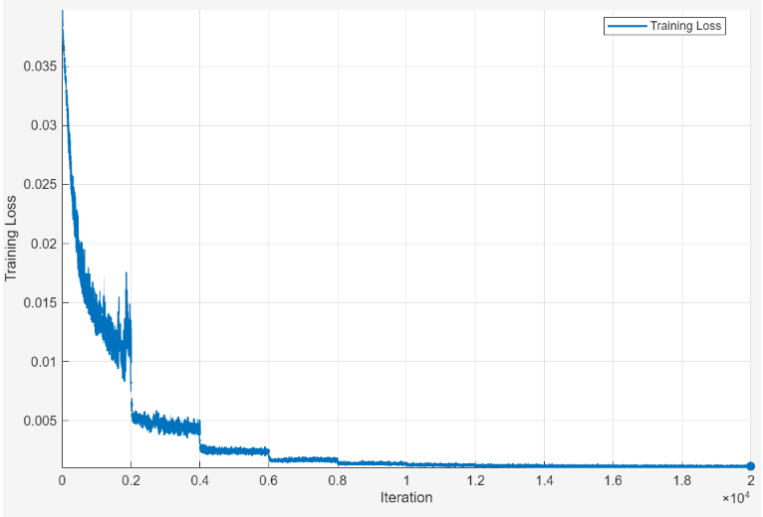

Figure 6.6: Training loss plot during offline training.

6.3.3 Simulation Results

The DL-based P-IPIC channel estimation algorithm works as follows: during the cost function maximization, the CDDPM matrix is obtained using FNN predictions. After that, before computing the channel gains vector, in order to maintain high estimation accuracy, the CDDPM column corresponding to the detected path is evalueted according to Equation (4.7) and stored. Hence, the maximum number of evaluations of Equation (4.7), considerng both *search phase* and *refinement phase*, is $2P_{\text{max}}$. Therefore, the latency of the P-IPIC algorithm is reduced since maximization of the cost function is performed using DL predictions.

The performance of the DL-based P-IPIC channel estimation algorithm are evaluated considering the high-IPI scenario described in Section (5.4.2).

Channel Estimation Performance

The accuracy of channel estimates is measured through NMSE as a function of PSNR. As shown in Figure 6.7, the DL-based P-IPIC exhibits lower performance due to the inaccurate computation of the CDDPM as a result of the finite learning capabilities of the two neural networks. In particular, at low and mid PSNR the DL-based P-IPIC shows a NMSE 1 dB lower with respect to the DDIPIC algorithm. However, at high PSNR, the performance gap increases to 3-4 dB. This phenomenon occurs since, at higher PSNR values, computational inaccuracies leads to higher residual IPI that is interpreted as small amplitudes paths. Thus, the estimation accuracy is reduced.

Number of Detected Paths

The detection capabilities of the DL-based P-IPIC are shown in Figure 6.8. For the DDIPIC and P-IPIC algorithms, the DL-based P-IPIC shows a similar behavior. In particular, at low PSNR the average number of detected paths is lower with respect to the true value and becomes equal to the true value at PSNR $= 26$ dB. At high PSNR, the DL-based P-IPIC overestimates the number of paths by $1 - 1.5$. This is consistent with the increasing gap between the non DL-based solutions and DL-based one in the NMSE performance and is due to the limited capabilities of the network to learn the columns of the TF CDDPM.

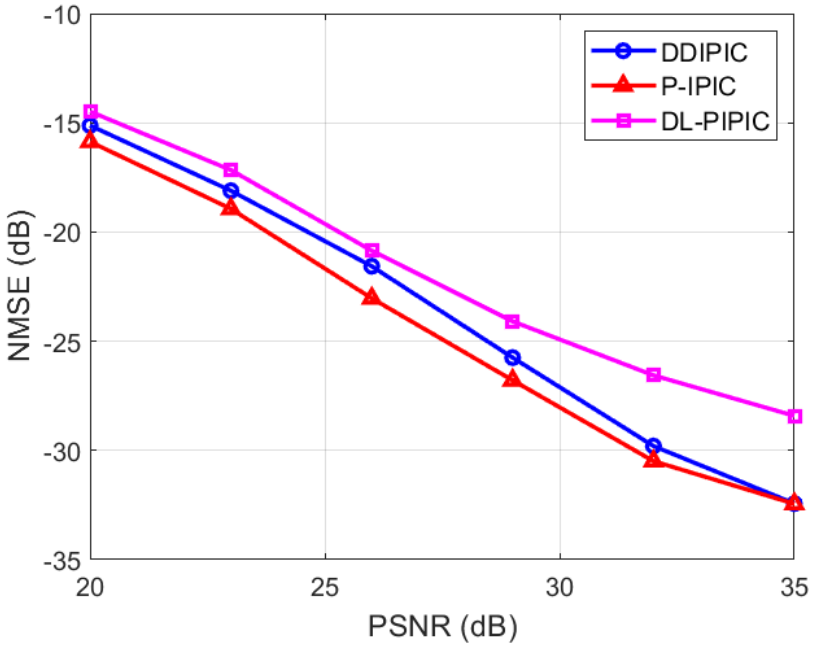

Figure 6.7: Normalized MSE as a function of pilot SNR.

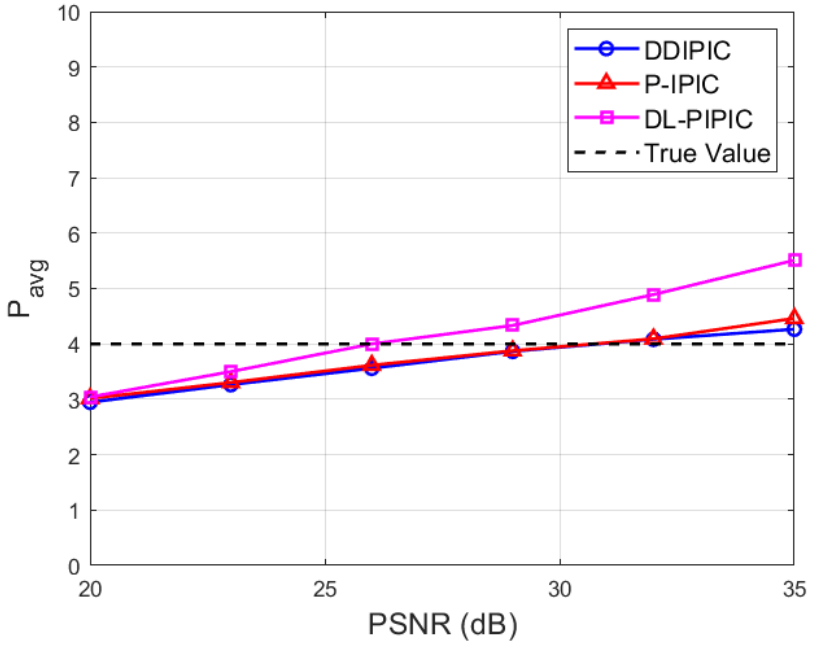

Figure 6.8: Average number of detected paths as a function of pilot SNR.

Chapter 7

Conclusions

7.1 Summary and final discussion

In this thesis, the fractional channel estimation problem in future 6G OTFS receivers is addressed. In the first part, a variant of the state-of-the-art DDIPIC algorithm, named P-IPIC, is presented. This variant is based on residual-based cost function maximization during the *search phase* and a signle global refinement aimed to reduce the computational burden of the DDIPIC algorithm, that, on the other side, performs several refinements procedures. This is a key advantage of the proposed approach since low-complexity and low-latency algorithms are needed to make 6G able to offer the desired perfomance and to increase OTFS practicabilty towards the implementation of 6G systems. Moreover, the P-IPIC algorithm is shown to exhibit better performance in both low- and high-IPI regimes than the DDIPIC. In particular the performance gain in low-IPI regime is almost zero at low PSNR while it rises up to 9 dB at PSNR $= 35$ dB. On the other side, in the high-IPI regime the performance gain is roughly $1 - 2$ dB and becomes zero at PSNR $= 35$ dB due to error propagation triggered by higher residual IPI.

Since complexity reduction is the one of the main challenges in OTFS systems, deep learning solutions can be adopted to face this problem. Hence, in the second part, a DL-based approach is developed in order to further reduce the latency of the proposed channel estimation algorithm. A DL-based architecture including two fully connected neural networks is used to learn real and imaginary parts of the TF CDDPM column. In this way, the DL-based P-IPIC algorithm exhibits, as expected, lower performance but at a drastically reduced latency and computational complexity. The performance loss due to function approximation is roughly 1 dB at low PSNR and $3-4$ dB at higher PSNR values. In front of this latency and complexity reduction, this performance loss is more then acceptable.

In conclusion, in this work a DL-based channel estimation algorithm for practical OTFS channel estimation, aimed to make OTFS more practical, is developed, achiev-

ing good performance in channel estimation with fractional parameters.

7.2 Future developments

As future works, different learning architectures can be studied to further reduce the complexity and latency in channel estimation and enhance accuracy. Since in this work it has been shown that a single global refinement can be used to increase performance and the *refinement phase* of the P-IPIC algorithm requires high computational complexity due to the computation of the observation-based cost function, a DL-based solution can be investigated to replace the iterative *refinement phase* with a single neural network capable of reproducing a refined version of channel parameters. In particular, the network may receive as inputs the estimated channel parameters obtained in the *search phase* together with the residue vector and, correlating those informations, it provides as output the refined channel parameters. In this way, the global iterative refinement procedure is replaced by a network able to provide in one-shot the refined estimates. Figure 7.1 shows a scheme of this architecture.

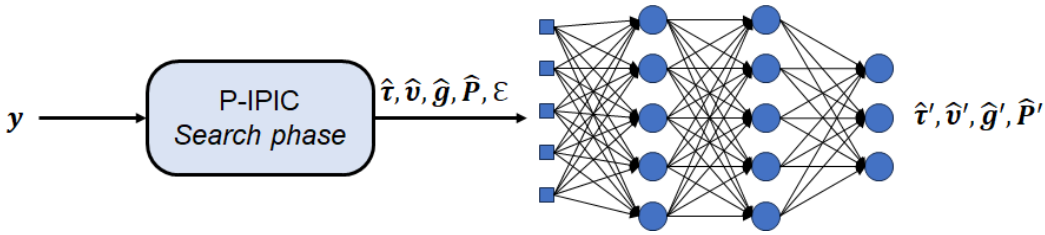

Figure 7.1: DL-based refinement procedure, a future perspective.

Moreover, adaptive solutions including continuous learning algorithms can be investigated to make channel estimation algorithms robust to nonstationary noise sources (or non-white noise sources) and interferers. Hence, channel estimation algorithms can run in different environmental conditions and adapt to guarantee high estimation accuracy. Other solutions may include parallel path estimation, in contrast with iterative solutions, and self-adaptive threshold to enhance detection capabilities of the channel estimation algorithms.

This future developments may bring this kind of channel estimation algorithms to be used in Integrated Sensing and Communication (ISAC) systems for both channel estimation and radar parameter estimation with multi-target detection. This is due to the fact that, if the capabilities in detecting the correct number of paths (or targets, in analogy) is increased, the same channel estimation algorithm can be used for multi-target detection in radar applications, localization and for sensing the channel.

Bibliography

[1] Yi Hong, Tharaj Thaj, and Emanuele Viterbo. *Delay-Doppler Communications: Principles and Applications*. English. Publisher Copyright: © 2022 Elsevier Inc. All rights reserved. Netherlands: Elsevier, 2022. ISBN: 9780323859660. DOI: 10.1016/C2020-0-01791-3.

[2] Marco Giordani et al. "Toward 6G Networks: Use Cases and Technologies". In: *IEEE Communications Magazine* 58.3 (2020), pp. 55–61. DOI: 10.1109/MCOM.001.1900411.

[3] Zhengquan Zhang et al. "6G Wireless Networks: Vision, Requirements, Architecture, and Key Technologies". In: *IEEE Vehicular Technology Magazine* 14.3 (2019), pp. 28–41. DOI: 10.1109/MVT.2019.2921208.

[4] Yuhong Huang et al. "6G mobile network requirements and technical feasibility study". In: *China Communications* 19.6 (2022), pp. 123–136. DOI: 10.23919/JCC.2022.06.010.

[5] Yu Zhou et al. *Overview and Performance Analysis of Various Waveforms in High Mobility Scenarios*. 2023. arXiv: 2302.14224 [eess.SP].

[6] R. Hadani et al. "Orthogonal Time Frequency Space Modulation". In: *2017 IEEE Wireless Communications and Networking Conference (WCNC)*. 2017, pp. 1–6. DOI: 10.1109/WCNC.2017.7925924.

[7] Ronny Hadani and Anton Monk. *OTFS: A New Generation of Modulation Addressing the Challenges of 5G*. 2018. arXiv: 1802.02623 [cs.IT].

[8] Imran Ali Khan and Saif Khan Mohammed. *Low Complexity Channel Estimation for OTFS Modulation with Fractional Delay and Doppler*. 2021. arXiv: 2111.06009 [cs.IT].

[9] Olivia Zacharia and M Vani Devi. "Fractional Delay and Doppler Estimation for OTFS Based ISAC Systems". In: *2023 IEEE Wireless Communications and Networking Conference (WCNC)*. 2023, pp. 1–6. DOI: 10.1109/WCNC55385.2023.10118873.

[10] P. Raviteja, Khoa T. Phan, and Yi Hong. "Embedded Pilot-Aided Channel Estimation for OTFS in Delay–Doppler Channels". In: *IEEE Transactions on Vehicular Technology* 68.5 (2019), pp. 4906–4917. DOI: 10.1109/TVT.2019.2906357.

[11] Sai Pradeep Muppaneni, Sandesh Rao Mattu, and A. Chockalingam. "Channel and Radar Parameter Estimation With Fractional Delay-Doppler Using OTFS". In: *IEEE Communications Letters* 27.5 (2023), pp. 1392–1396. DOI: 10.1109/LCOMM.2023.3251578.

[12] P. Raviteja et al. "Interference Cancellation and Iterative Detection for Orthogonal Time Frequency Space Modulation". In: *IEEE Transactions on Wireless Communications* 17.10 (2018), pp. 6501–6515. DOI: 10.1109/TWC.2018.2860011.

[13] P. Raviteja et al. "Low-complexity iterative detection for orthogonal time frequency space modulation". In: *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. 2018, pp. 1–6. DOI: 10.1109/WCNC.2018.8377159.

[14] Shashank Tiwari, Suvra Sekhar Das, and Vivek Rangamgari. "Low complexity LMMSE Receiver for OTFS". In: *IEEE Communications Letters* 23.12 (2019), pp. 2205–2209. DOI: 10.1109/LCOMM.2019.2945564.

[15] G. D. Surabhi and A. Chockalingam. "Low-Complexity Linear Equalization for OTFS Modulation". In: *IEEE Communications Letters* 24.2 (2020), pp. 330–334. DOI: 10.1109/LCOMM.2019.2956709.

[16] Wenqian Shen et al. "Channel Estimation for Orthogonal Time Frequency Space (OTFS) Massive MIMO". In: *IEEE Transactions on Signal Processing* 67.16 (2019), pp. 4204–4217. DOI: 10.1109/TSP.2019.2919411.

[17] Vineetha Yogesh, Sandesh Rao Mattu, and A. Chockalingam. "Low-Complexity Delay-Doppler Channel Estimation in Discrete Zak Transform Based OTFS". In: *IEEE Communications Letters* 28.3 (2024), pp. 672–676. DOI: 10.1109/LCOMM.2024.3351685.

[18] Lei Zhao, Wen-Jing Gao, and Wenbin Guo. "Sparse Bayesian Learning of Delay-Doppler Channel for OTFS System". In: *IEEE Communications Letters* 24.12 (2020), pp. 2766–2769. DOI: 10.1109/LCOMM.2020.3021120.

[19] Zhiqiang Wei et al. "Off-Grid Channel Estimation With Sparse Bayesian Learning for OTFS Systems". In: *IEEE Transactions on Wireless Communications* 21.9 (2022), pp. 7407–7426. DOI: 10.1109/TWC.2022.3158616.

[20] Huan-Tang Sheng and Wen-Rong Wu. "Time-Frequency Domain Channel Estimation for OTFS Systems". In: *IEEE Transactions on Wireless Communications* 23.2 (2024), pp. 937–948. DOI: 10.1109/TWC.2023.3283578.

[21] L. Gaudio et al. "On the Effectiveness of OTFS for Joint Radar Parameter Estimation and Communication". In: *IEEE Transactions on Wireless Communications* 19.9 (Sept. 2020). DOI: 10.1109/TWC.2020.2998583.

[22] M. Marchese et al. "Progressive Inter-Path Interference Cancellation Algorithm for Channel Estimation Using Orthogonal Time–Frequency Space". In: *Sensors* 24.4414 (2024). DOI: 10.3390/s24134414. URL: https://doi.org/10.3390/s24134414.

[23] Linglong Dai et al. "Deep Learning for Wireless Communications: An Emerging Interdisciplinary Paradigm". In: *IEEE Wireless Communications* 27.4 (2020), pp. 133–139. DOI: 10.1109/MWC.001.1900491.

[24] S. Haykin. *Neural Networks and Learning Machines*. Third. McMaster University, Hamilton: Pearson Education, Inc., 2009.

[25] Ashwitha Naikoti and A. Chockalingam. "A DNN-based OTFS Transceiver with Delay-Doppler Channel Training and IQI Compensation". In: *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2021, pp. 628–634. DOI: 10.1109/PIMRC50174.2021.9569634.

[26] Ashwitha Naikoti and A. Chockalingam. "Low-complexity Delay-Doppler Symbol DNN for OTFS Signal Detection". In: *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. 2021, pp. 1–6. DOI: 10.1109/VTC2021-Spring51267.2021.9448630.

[27] Qingyu Li et al. "Residual Learning based Channel Estimation for OTFS system". In: *2022 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*. 2022, pp. 275–280. DOI: 10.1109/ICCCWorkshops55477.2022.9896637.

[28] Sandesh Rao Mattu and A. Chockalingam. "Learning based Delay-Doppler Channel Estimation with Interleaved Pilots in OTFS". In: *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*. 2022, pp. 1–6. DOI: 10.1109/VTC2022-Fall57202.2022.10012974.

[29] Xiaoqi Zhang et al. "Deep Learning with a Self-Adaptive Threshold for OTFS Channel Estimation". In: *2022 International Symposium on Wireless Communication Systems (ISWCS)*. 2022, pp. 1–5. DOI: 10.1109/ISWCS56560.2022.9940260.

[30] Xiaoqi Zhang, Weijie Yuan, and Chang Liu. "Deep Residual Learning for OTFS Channel Estimation with Arbitrary Noise". In: *2022 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*. 2022, pp. 320–324. DOI: 10.1109/ICCCWorkshops55477.2022.9896721.

[31] Yuan Gao et al. “Channel Estimation for MIMO-OTFS Satellite Communication: a Deep Learning-Based Approach”. In: *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*. 2024, pp. 1–6. DOI: 10.1109/ICCCN61486.2024.10637519.

[32] Sandesh Rao Mattu and A. Chockalingam. “Learning in Time-Frequency Domain for Fractional Delay-Doppler Channel Estimation in OTFS”. In: *IEEE Wireless Communications Letters* 13.5 (2024), pp. 1245–1249. DOI: 10.1109/LWC.2024.3367112.

Appendix A

MATLAB code

In this appendix some MATLAB code used for simulations is reported. Some custom functions used in the code are not reported with their own code since are straightforward implementations.

A.1 DDIPIC Algorithm

```
function [H_hat, P_hat, tau_hat, nu_hat, gi_hat] =
   DDIPIC_ChEst(x, tau_max, nu_max, M, N, T, Df, P_max, eps,
   max_iter, eps_tau, eps_nu, m_tau, n_nu, y)

% DD Parameters
delay_res = 1 / (M * Df);
Doppler_res = 1 / (N * T);
spf = M * N;

% Inizialization
CDDPM = zeros(spf, P_max);

% Coarse Estimate Search Space
L = ceil(tau_max / delay_res);
tau_int = zeros(1, L + 1);
for l = 0 : L
    tau_int(1, l + 1) = l * delay_res;
end
K = ceil(nu_max / Doppler_res);
nu_int = zeros(1, 2 * K + 1);
for k = -K : K
    nu_int(1, k + K + 1) = k * Doppler_res;
```

```
21 end
22
23 % Fine Estimate Search Space Parameters
24 N_tau = 2 * floor(m_tau / 2) + 1;
25 N_nu = 2 * floor(n_nu / 2) + 1;
26
27 % Estimated DDs Vectors and Channel Gains
28 tau_hat = zeros(1, P_max);
29 nu_hat = zeros(1, P_max);
30 gi_hat = zeros(1, P_max);
31
32 % DDIPIC Algorithm
33 for i = 1 : P_max
34     R = CDDPM;
35
36     % Coarse Estimate
37     [CDDPM, tau_coarse, nu_coarse] = argmaxcostf_ddipic(R, y,
          tau_int, nu_int, x, M, N, T, Df, i);
38
39     % Fine Estimate
40     tau_s = tau_coarse;
41     nu_s = nu_coarse;
42
43     for s_fine = 1 : max_iter
44
45         % DD Search Widths
46         W_tau = delay_res / m_tau^(s_fine - 1);
47         W_nu = Doppler_res / n_nu^(s_fine - 1);
48
49         % Search Space
50         if tau_s == 0
51            Gamma = 0 : (N_tau - 1) / 2;
52         elseif tau_s > 0
53                Gamma = -(N_tau - 1) / 2 : (N_tau - 1) / 2;
54         end
55         Lambda = -(N_nu - 1) / 2 : (N_nu - 1) / 2;
56         tau_frac = W_tau * Gamma + tau_s;
57         tau_frac = tau_frac(tau_frac >= 0);
58         nu_frac = W_nu * Lambda + nu_s;
59
60         % Cost Function Maximization
61         [CDDPM, tau_s1, nu_s1] = argmaxcostf_ddipic(R, y,
              tau_frac, nu_frac, x, M, N, T, Df, i);
```

```
62
63          % Stopping Criterion for Fine Estimate Loop
64          if s_fine == 1
65              tau_s = tau_s1;
66              nu_s = nu_s1;
67          elseif s_fine > 1
68              if abs(tau_s1 - tau_s) < eps_tau && abs(nu_s1 -
                    nu_s) < eps_nu || s_fine == max_iter
69                  tau_fine = tau_s1;
70                  nu_fine = nu_s1;
71                  break
72              elseif s_fine < max_iter
73                  tau_s = tau_s1;
74                  nu_s = nu_s1;
75              end
76          end
77      end
78
79      % Update Estimated DD Vectors
80      tau_hat(1, i) = tau_fine;
81      nu_hat(1, i) = nu_fine;
82
83      % Channel Gains Vector Computation
84      gi_hat(1, 1 : i) = (CDDPM(:, 1 : i)' * CDDPM(:, 1 :
            i))^-1 * CDDPM(:, 1 : i)' * y;
85
86      % Stopping Criterion for DDIPIC
87      Res_i = y - CDDPM(:, 1 : i) * gi_hat(1, 1 : i).';
88      if i > 1
89              if norm(Res_i) < eps|| i == P_max
90                  P_hat = i;
91                  break
92              else
93                  % Refinement Step
94                  for i_ref = 1 : i
95                      R = CDDPM;
96                      R(:, i_ref) = zeros(spf, 1);
97
98                      % (Refined) Coarse Estimate
99                      [CDDPM, tau_coarse, nu_coarse] =
                            argmaxcostf_ref(R, y, tau_int, nu_int,
                            x, M, N, T, Df, i_ref, i);
100
```

```
101                     % (Refined) Fine Estimate
102                     tau_s = tau_coarse;
103                     nu_s = nu_coarse;
104
105                     for s_fine = 1 : max_iter
106
107                         % DD Search Widths
108                         W_tau = delay_res / m_tau^(s_fine - 1);
109                         W_nu = Doppler_res / n_nu^(s_fine - 1);
110
111                         % Search Space
112                         if tau_s == 0
113                             Gamma = 0 : (N_tau - 1) / 2;
114                         elseif tau_s > 0
115                             Gamma = -(N_tau - 1) / 2 : (N_tau
                                    - 1) / 2;
116                         end
117                         Lambda = -(N_nu - 1) / 2 : (N_nu - 1)
                                / 2;
118                         tau_frac = W_tau * Gamma + tau_s;
119                         tau_frac = tau_frac(tau_frac >= 0);
120                         nu_frac = W_nu * Lambda + nu_s;
121
122                         % Cost Function Maximization
123                         [CDDPM, tau_s1, nu_s1] =
                                argmaxcostf_ref(R, y, tau_frac,
                                nu_frac, x, M, N, T, Df, i_ref, i);
124
125                         % Stopping Criterion for Fine Estimate
                                Loop
126                         if s_fine == 1
127                             tau_s = tau_s1;
128                             nu_s = nu_s1;
129                         elseif s_fine > 1
130                             if abs(tau_s1 - tau_s) < eps_tau
                                    && abs(nu_s1 - nu_s) < eps_nu
                                    || s_fine == max_iter
131                                 tau_fine = tau_s1;
132                                 nu_fine = nu_s1;
133                                 break
134                             elseif s_fine < max_iter
135                                 tau_s = tau_s1;
136                                 nu_s = nu_s1;
```

```
                            end
                        end
                    end

                    % Update Estimated DD Vectors
                    tau_hat(1, i_ref) = tau_fine;
                    nu_hat(1, i_ref) = nu_fine;
                end
            end
        end
    end

% Generate Estimated DD-Domain Channel Matrix
H_hat = genDDchannelmatrix(P_hat, gi_hat, tau_hat, nu_hat, M,
    N, T, Df);

end
```

A.2 P-IPIC Algorithm

```
function [H_hat, P_hat, tau_hat, nu_hat, gi_hat] =
    PIPIC_ChEst(x, tau_max, nu_max, M, N, T, Df, P_max, eps,
    max_iter, eps_tau, eps_nu, m_tau, n_nu, y)

% DD Parameters
delay_res = 1 / (M * Df);
Doppler_res = 1 / (N * T);
spf = M * N;

% Inizialization
CDDPM = zeros(spf, P_max);

% Coarse Estimate Search Space
L = ceil(tau_max / delay_res);
tau_int = zeros(1, L + 1);
for l = 0 : L
    tau_int(1, l + 1) = l * delay_res;
end
K = ceil(nu_max / Doppler_res);
nu_int = zeros(1, 2 * K + 1);
for k = -K : K
```

```
20      nu_int(1, k + K + 1) = k * Doppler_res;
21  end
22
23  % Fine Estimate Search Space Parameters
24  N_tau = 2 * floor(m_tau / 2) + 1;
25  N_nu = 2 * floor(n_nu / 2) + 1;
26
27  % Estimated DDs Vectors and Channel Gains
28  tau_hat = zeros(1, P_max);
29  nu_hat = zeros(1, P_max);
30  gi_hat = zeros(1, P_max);
31
32  % Residue Initialization
33  Res_i = y;
34
35  % P-IPIC Algorithm
36  for i = 1 : P_max
37      R = CDDPM;
38
39      % Coarse Estimate
40      [CDDPM, tau_coarse, nu_coarse] = argmaxcostf_pipic(R,
            Res_i, tau_int, nu_int, x, M, N, T, Df, i);
41
42      % Fine Estimate
43      tau_s = tau_coarse;
44      nu_s = nu_coarse;
45
46      for s_fine = 1 : max_iter
47
48          % DD Search Widths
49          W_tau = delay_res / m_tau^(s_fine - 1);
50          W_nu = Doppler_res / n_nu^(s_fine - 1);
51
52          % Search Space
53          if tau_s == 0
54             Gamma = 0 : (N_tau - 1) / 2;
55          elseif tau_s > 0
56                 Gamma = -(N_tau - 1) / 2 : (N_tau - 1) / 2;
57          end
58          Lambda = -(N_nu - 1) / 2 : (N_nu - 1) / 2;
59          tau_frac = W_tau * Gamma + tau_s;
60          tau_frac = tau_frac(tau_frac >= 0);
61          nu_frac = W_nu * Lambda + nu_s;
```

```
62
63          % Cost Function Maximization
64          [CDDPM, tau_s1, nu_s1] = argmaxcostf_pipic(R, Res_i,
               tau_frac, nu_frac, x, M, N, T, Df, i);
65
66          % Stopping Criterion for Fine Estimate Loop
67          if s_fine == 1
68              tau_s = tau_s1;
69              nu_s = nu_s1;
70          elseif s_fine > 1
71              if abs(tau_s1 - tau_s) < eps_tau && abs(nu_s1 -
                   nu_s) < eps_nu || s_fine == max_iter
72                  tau_fine = tau_s1;
73                  nu_fine = nu_s1;
74                  break
75              elseif s_fine < max_iter
76                  tau_s = tau_s1;
77                  nu_s = nu_s1;
78              end
79          end
80      end
81
82      % Update Estimated DD Vectors
83      tau_hat(1, i) = tau_fine;
84      nu_hat(1, i) = nu_fine;
85
86      % Channel Gains Vector Computation
87      gi_hat(1, 1:i) = (CDDPM(:, 1:i)' * CDDPM(:, 1:i) +
           eye(i))^-1 * CDDPM(:, 1:i)' * y;
88
89      % Stopping Criterion for P-IPIC
90      Res_i = y - CDDPM(:, 1:i) * gi_hat(1, 1:i).';
91      if i > 1
92          if norm(Res_i) < eps || i == P_max
93              P_hat = i;
94              break
95          end
96      end
97  end
98
99  % Refinement Procedure
100 for i_ref = 1 : P_hat
101     R = CDDPM;
```

```
102     R(:, i_ref) = zeros(spf, 1);
103
104     % (Refined) Coarse Estimate
105     [CDDPM, tau_coarse, nu_coarse] = argmaxcostf_ref_pipic(R,
           y, tau_int, nu_int, x, M, N, T, Df, i_ref, P_hat);
106
107     % (Refined) Fine Estimate
108     tau_s = tau_coarse;
109     nu_s = nu_coarse;
110
111     for s_fine = 1 : max_iter
112
113         % DD Search Widths
114         W_tau = delay_res  / m_tau^(s_fine - 1);
115         W_nu = Doppler_res / n_nu^(s_fine - 1);
116
117         % Search Space
118         if tau_s == 0
119             Gamma = 0 : (N_tau - 1) / 2;
120         elseif tau_s > 0
121             Gamma = -(N_tau - 1) / 2 : (N_tau - 1) / 2;
122         end
123         Lambda = -(N_nu - 1) / 2 : (N_nu - 1) / 2;
124         tau_frac = W_tau * Gamma + tau_s;
125         tau_frac = tau_frac(tau_frac >= 0);
126         nu_frac = W_nu * Lambda + nu_s;
127
128         % Cost Function Maximization
129         [CDDPM, tau_s1, nu_s1] = argmaxcostf_ref_pipic(R, y,
              tau_frac, nu_frac, x, M, N, T, Df, i_ref, P_hat);
130
131         % Stopping Criterion for Fine Estimate Loop
132         if s_fine == 1
133             tau_s = tau_s1;
134             nu_s = nu_s1;
135         elseif s_fine > 1
136             if abs(tau_s1 - tau_s) < eps_tau && abs(nu_s1 -
                   nu_s) < eps_nu || s_fine == max_iter
137                 tau_fine = tau_s1;
138                 nu_fine = nu_s1;
139                 break
140             elseif s_fine < max_iter
141                 tau_s = tau_s1;
```

```
142                 nu_s = nu_s1;
143             end
144         end
145     end
146
147     % Update Estimated DD Vectors
148     tau_hat(1, i_ref) = tau_fine;
149     nu_hat(1, i_ref) = nu_fine;
150
151     % Channel Gains Vector Computation
152     gi_hat(1, 1:i_ref) = (CDDPM(:, 1:i_ref)' * CDDPM(:,
           1:i_ref) + eye(i_ref))^-1 * CDDPM(:, 1:i_ref)' * y;
153
154     %Residue Computation
155     Res_i = y - CDDPM(:, 1:i_ref) * gi_hat(1, 1:i_ref).';
156     if i_ref > 1
157         if norm(Res_i) < eps
158             P_hat = i_ref;
159             break
160         end
161     end
162 end
163
164 % Generate Estimated DD-Domain Channel Matrix
165 H_hat = genDDchannelmatrix(P_hat, gi_hat, tau_hat, nu_hat, M,
       N, T, Df);
166
167 end
```