# Wavelet-based spatial audio framework

From Ambisonics to wavelets: a novel approach to spatial audio.


## Author: Davide Scaini



TESI DOCTORAL UPF / year of the thesis: 2019



THESIS SUPERVISORS
Daniel Arteaga Barriel
Ricardo Baeza-Yates

Department of Information and Communication Technologies

Universitat Pompeu Fabra Barcelona

In theory there is no difference
between theory and practice;
in practice there is.

# Acknowledgments

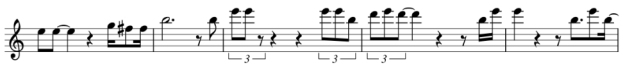
Manu for being part of this journey, with her personality and the desire to experiment every day how to live our thing. We have plans.

Dani, for his guidance and persistence. Someone once said that only two things are infinite, the Universe and Dani's patience. I have extensive experimental data for the second. I owe you a lot. More than I can express.

Barcelona Media for giving me the opportunity to start a new life in a different field in a wonderful city.

Thanks to Ricardo for the thorough review of this manuscript.

Dolby, in the persons of Toni and Claus, for believing in me and gently pushing me towards this goal. A sincere and happy "thank you!" is in order. I am very grateful.



# Abstract

Ambisonics is a complete theory for spatial audio whose building blocks are the spherical harmonics. Some of the drawbacks of low order Ambisonics, like poor source directivity and small sweet-spot, are directly related to the properties of spherical harmonics. In this thesis we illustrate a novel spatial audio framework similar in spirit to Ambisonics that replaces the spherical harmonics by an alternative set of functions with compact support: the spherical wavelets. We develop a complete audio chain from encoding to decoding, using discrete spherical wavelets built on a multiresolution mesh. We show how the wavelet family and the decoding matrices to loudspeakers can be generated via numerical optimization. In particular, we present a decoding algorithm optimizing acoustic and psychoacoustic parameters that can generate decoding matrices to irregular layouts for both Ambisonics and the new wavelet format. This audio workflow is directly compared with Ambisonics.



# Resum

Ambisonics és una teoria completa d'àudio espacial construïda a partir dels harmònics esfèrics. Alguns dels inconvenients d'Ambisonics de baix ordre, com ara una localització pobra i una àrea petita d'escolta òptima, estan directament relacionats amb les propietats dels harmònics esfèrics. En aquesta tesi presentem un nou formalisme d'àudio espacial basat en Ambisonics substituint però els harmònics esfèrics per les ondetes esfèriques. Desenvolupem una cadena d'àudio completa, des de la codificació fins a la descodificació, a través de l'ús de ondetes discretes construïdes en una malla de multirresolució. Mostrem com es pot generar la família de ondetes i les matrius de descodificació a altaveus mitjançant una optimització numèrica. Presentem un algorisme de descodificació que pot generar matrius de descodificació a conjunts irregulars d'altaveus tant per a Ambisonics com per al nou format basat en ondetes. Finalment, comparem aquest nou formalisme d'àudio amb Ambisonics.

# Preface

It was 2012 when I started my Ph.D. in Information and Communication Technologies at the University Pompeu Fabra in Barcelona. At that time Dolby and DTS were launching their solutions for object-based audio, with Dolby Atmos and DTS X. At the same time Auro 3D took the opposite direction, making its bet on a new channel based format. The Academia was (and still is) more focused on sound-field reconstruction methods. Virtual Reality (VR) was not yet a trend, with Facebook and Google leading the "democratization" of 3D video and ultimately 3D audio, making use of Ambisonics for binaural rendering. (The Google Cardboard was launched in June 2014). This was finally the point when the (mass-scale) industry met academia.

Object-based formats are generic representations of sound scenes, and are a powerful tool for 3D soundscape creation. Given their rendering-agnostic construction, they can be plugged to almost any spatial audio rendering technology, from amplitude panners to sound-field reconstruction methods. Given this scenario, we felt that there was still room for improvement in existing technologies, making them more robust and easy to use, from an acoustic point of view.

We started with Ambisonics and identified that the stage of decoding to speakers was still a pain point for the public wanting to experiment with it. We decided to make a new decoder, that built on well known psychoacoustic principles, but at the same time bent slightly the dogmas of Ambisonics, like constant $r_E$ and null $r_E$ transverse component, to get a better sounding rendition. And we released it as open-source software (2013).

Approximately at the same time Aaron Heller released his Ambisonics Toolbok (2014), that is a collection of tools to easily generate Ambisonics decoders.

After almost 40 years from the creation of Ambisonics by Michael Gerzon, the researchers were (and are) still working some of its edges.

Nevertheless, this research was not addressing the core drawbacks of Ambisonics, and we started looking for some alternatives to Spherical Harmonics. It was Pau Gargallo that suggested to look into spherical wavelets, since he was familiar with Schröder and Sweldens' work. As a particle physicist I was not familiar with wavelets at all, and I started wandering into this new (for me) world. We started from the very early works of Haar, Gabor, Morlet, Meyer, Mallat all the way to Daubechies, then moved to the spherical manifold with the works of Wiaux and McEwans, looking at different sampling theorems on the sphere, and then Christian Lessig with his SOHO wavelets... and finally we landed where everything started (at least for us) with Schröder and Sweldens.

The final concept we developed allows to encode sound sources to a cloud of points and to reduce (or recover) the dimensionality of the cloud at will. The spatial downsampling is implemented as a linear transformation that can be fully reverted. This construction allows for different coexisting spatial representations, that can scale based on different requirements, for example transmission bandwidth or the complexity of the destination playback system. We call this construction a "framework", that is used to generate actual audio formats. In this thesis we illustrate the general framework and one special format designed to be compared with and evaluated against Ambisonics.

Interestingly, for the decoding to speakers of this new format, the same principles used for Ambisonics decoding apply.

We then tried to push the idea further, by generating our own spatial-audio-oriented wavelets. The idea was to numerically optimize the wavelets for some observables, e.g. pressure preservation across the down/upsampling process. This literally took years...

It has been a long journey with many dead-ends, but we think we found something interesting and new in the spatial-audio field, that may inspire old and young researchers.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# INTRODUCTION

This thesis is a first point of connection between two worlds, the world of *wavelets*, which is related to time-frequency analysis and signal processing, and the world of *spatial audio*, which embraces sound perception, sound recording, encoding and reproduction.

Section 1.1 briefly describes what is spatial audio and gives some context about the available technologies. Section 1.3 outlines the motivations for this thesis.

## 1.1 What is Spatial Audio?

Among the five senses that humans can experience, *hearing* or audition is the sense of sound perception. Humans are able to identify the location of a sound in *direction*, *distance* and *size*. *Spatial audio* refers to the set of tools, technologies and theories for *creation* or *recreation* of a subjective sound scene, that has to produce all the *spatial characteristics* of a sound located in a 2D/3D space: **direction**, **distance**, **size**.

It is possible to classify the techniques to (re)create an auditory scene (2D or 3D) in three categories:

1. *Discrete panning techniques* (e.g. VBAP, ABAP, VBIP, ABIP): the known apparent direction of the source is used to feed a limited number of loudspeakers.

2. *Sound field reconstruction methods* (e.g. Ambisonics, Wave Field Synthesis): the intent is to control the acoustical variables of the sound field (pressure, velocity) in the listening space.

3. *Head-related stereophony* (binaural, transaural): the aim is to measure (binaural recording) or (re)produce (binaural synthesis) the acoustic pressure at the ears of the listener.

Besides the underlying theory of each technology, the spatial audio techniques can be also classified by analyzing how the whole encoding/decoding pipeline is structured:

- *Channel-based*: the whole encoding/decoding and recording/reproduction is based on a specific channel layout, e.g. 2.0, 5.1, 7.1, ..., Auro3D, Hamasaki 22.2.

- *Layout-independent* (channel-agnostic): the recording and encoding format is independent from the reproduction layout (includes sound field reconstruction methods and object-based formats).

Malham [Malham, 1999] gives an interesting perspective on the existing (at that time) surround sound systems, but also gives two criteria that can be applied to any existing or future technology: the ideas of homogeneous and coherent sound reproduction systems. Quoting from [Malham, 1999]:

> "An *homogeneous* sound reproduction system is defined as one in which no direction is preferentially treated. A *coherent* system as one in which the image remains stable if the listener changes position within it, though the image may change as a natural soundfield does."

With this set of properties we can start categorize the existing technologies. For example: VBAP is a channel-based discrete panning technique, which is not coherent (e.g. the apparent source size depends on the position of the source) and in general not homogeneous. Ambisonics is a theory that aims at reconstructing the sound field, is layout-independent, is coherent and homogeneous.

Why there is this variety of techniques? Why not one method that works in all possible conditions? Each technology has its strengths and weaknesses and a specific area of application.

## 1.2 Benefits and Limitations

In the following, we will give a non exhaustive list of benefits and limitations of the mentioned techniques. We are aware that every point in these lists of *pros and cons* is a simplification, that is open for discussion and could generate many distinctions. The detailed description and dissection of each existing technique is out of the scope of this introduction.

**Discrete panning techniques** Pan-pot or stereo amplitude panning is a technique that, by independently changing the amplitude of a signal in a pair of loudspeakers, is able to generate a virtual sound source along the arc connecting the two speakers. The position of the virtual sound source depends on the difference in amplitude between the two speakers. Vector-Base Amplitude Panning (VBAP) is an extension of the classical stereo pan-pot to multi-speakers layouts (2D and 3D) and was invented by Ville Pulkki in 1997 [Pulkki, 1997]. Many variants of this technique are available, for example: Vector-Base Intensity Panning (VBIP), Distance-Based Amplitude Panning (DBAP) [Lossius et al., 2009].

+ *Simple* to implement.
+ The most *common*.
+ Easily to handle *non-uniform layouts*.
+ The experience holds decently also outside the sweet spot (true for layouts with many loudspeakers).

– Not a complete theory (e.g. recording in VBAP it is not possible).
– Not homogeneous and not coherent.

– Jump of the virtual source from speaker to speaker (apparent source size changes as a function of the virtual source position).

**Ambisonics** Ambisonics was invented by Michael Gerzon of the Oxford Mathematical Institute who developed the theoretical and practical aspects of the system in the early 1970s. Ambisonics comprises both encoding, recording and reproduction (decoding) techniques that can be used live or in studio to present a 2-dimensional (*planar*, or horizontal-only) or 3-dimensional (*periphonic*, or full-sphere) sound field. There are professional quality commercial microphones which can *directly* record in first order Ambisonics. In recent years higher order microphones have appeared, e.g. Eigenmike [mh acoustics LLC, 2019] and Zylia [Zylia, 2019], that enable the encoding of an Higher Order Ambisonics soundscape.

+ Nice *physical formulation*.

+ Aims to reproduce the *sound field*.

+ Is homogeneous and coherent.

+ Getting a lot of traction in Augmented/Virtual Reality (AR/VR) (for binaural reproduction over headphones), but has a minor role in the spatial audio industry as a whole.

– *Poor localization* at low orders.

– *Small sweet spot* at low orders.

– *Difficult* to handle *irregular layouts*.

– Not so common (reproduction over speakers limited to universities, research centres, ...).

– Sound source distance is not included in basic the theory, nevertheless there are implementations which enable distance encoding [Daniel, 2003].

**Wavefield Synthesis** Wavefield Synthesis (WFS) was introduced in the 1980s by Dr. A. J. Berkhout [Berkhout, 1988], a professor of seismics and acoustics at the Delft University of Technology. WFS focuses on the reproduction only and it is not possible to record directly in WFS. The typical WFS layouts are 2-dimensional, horizontal, and can be linear of circular. One advantage over Ambisonics is that the distance of the source is embedded in the theory.

+ Nice *physical formulation*.
+ Aims to reproduce the *sound field*.
+ Is homogeneous and coherent.
+ Sound source distance is embedded in the theory.
– Very limited diffusion, less common than Ambisonics (limited to few universities, research centres, ...), currently almost zero presence in the spatial audio industry.
– Expensive and difficult to set up (speaker layouts have to be specifically tailored to WFS).
– Computationally intense.

**Head-related stereophony** The techniques inside the realm of the head-related stereophony go from binaural (recording with in-ear microphones and reproduction over headphones), to transaural (reproduction over speakers), to generic 'virtualization' techniques. All these techniques exploit the information about head and outer-ear (pinna) shape to create the right pressure at the eardrums to simulate a sound in space.

+ (It can be) better than plain stereo over headphones.
+ Homogeneous (depending on the specific implementation).
+ Coherent for synthesized sound fields with head-tracking.
+ Cheap (in terms of production, transmission and exhibition).

– Extremely complex to get an experience that works for everyone with no tuning, training, ...

– Once produced, it is very difficult to modify and manipulate without destroying the experience.

**Object-based formats** In object based audio formats at each sound source is associated an audio track and some characteristics (metadata), that can be position, distance, size, and possibly more. Examples of commercial object-based formats are DTS® X and Dolby Atmos®. This kind of technologies are by construction layout-independent, since the rendering stage is (or can be) completely disconnected from the format itself. This means that one could choose different rendering techniques for the same (object-based) format.

+ Naturally handles any type of speaker layout (number and position of speakers).

+ Rendering is separate from the encoding and transmission stage, so in principle any type of rendering can be used.

– The number of channels to be transmitted/stored depends on the number of objects (audio sources).

– Since the rendering is done in real-time, the computational load scales with the number of objects.

**Differences in audio workflow** The typical audio workflow can be simplified in three stages: encoding, transmission and decoding (or playback). Each technology can attach a different meaning to each of these three stages. Here we intend the encoding (or decoding) as 'spatial encoding', i.e. the tools to encode a spatial signal into some format, and not the encoding to (or decoding from) a bitstream. In broad terms, and looking only to encoding and decoding, we will give some examples to clarify the differences in the audio workflow for different techniques.

In channel based formats the encoding involves the use of some panning law, that translates the spatial position of a source into gains of that signal for

each channel of the format. The panning law defines the format. The decoding stage is typically just the direct playback to speakers, since in a channel based format the channels correspond to actual speaker positions.

In channel agnostic (but not object based) formats the encoding format translates the three-dimensional position of the source to some other space, and the gains are the weights of the functions or filters that map the 3D space to the new one. In the decoding the process is reversed, and these 'abstract' channels regain their meaning in the space of positions.

In object based formats the role of encoding and decoding is essentially reversed with respect to the channel based formats. There is no spatial encoding, and the task of generating the gains for each speaker is given to the decoding stage, where a panning law is used at playback time to convert the object metadata into gains. Here the panning law does not define the format, and it is possible to use different panning laws for the same object based format, providing that the panning law 'understands' the object metadata.

## 1.3 Motivation

We have seen in Section 1.2 that there are several techniques for spatial audio and each of them has its purpose. The choice of the best technique is application dependent. An interesting evaluation of stereophony, Ambisonics and WFS in the context of spatial music can be found in [Bates, 2009]. There is no 'absolute best' solution that works for every condition and context. This thesis rises from this realization and the consequent question:

> Is it possible to build a theory for spatial audio that is channel agnostic, homogeneous and coherent, but also has good localization with few channels, easily handles irregular layouts and holds well when moving out of the sweet spot? In other words, *is it possible to build a theory that combines the best of channel-based and channel-agnostic worlds?*

The question already sets some requirements and a focus on the problem. A channel-agnostic format is preferred, since there are already many channel based formats that span a wide portion of the localization spectra, from the

low channel count and limited localization like stereo 2.0 and 5.1, to the very high channel count of Hamasaki 22.2. Moreover, the channel based formats are not flexible nor future proof. In the realm of channel agnostic formats we can find technologies like WFS and Ambisonics. WFS is not suited to irregular layouts and typically uses a large number of speakers. Ambisonics has a reasonable number of channels but it is not trivial to decode to irregular layouts and the sweet spot can be quite limited at low orders. The Ambisonics channels can be understood as a series expansion of the distribution of sources in terms of spherical harmonics (SH): the higher the order in the expansion, the more spatial detail and the bigger the sweet spot. Each Ambisonics coefficient corresponds to a SH function. The properties of SH are a key element in how Ambisonics works and sounds: all SH have significant non-vanishing support in all points of the sphere (except for a finite set of points), and moreover the SH are completely delocalized, meaning that it is not possible to assign an individual spatial location to any SH by itself. This implies that at low orders almost all speakers contribute significantly to create a virtual source. These properties of SH translate directly into the subjective characteristics of Ambisonics as a spatial audio format, which is often reported to be smooth, diffuse and immersive, but also confuse, imprecise, and delocalized.

In this thesis we develop a new spatial audio codification, similar in spirit to Ambisonics but replacing the SH by a different and more localized, set of functions: the *spherical wavelets*. The goal is to get better localization and a larger sweet spot with few channels, that can be easily decoded to irregular layouts.

In this context, we encounter a first gap to fill: successfully decode Ambisonics and Higher Order Ambisonics to irregular layouts. Part I focuses on this task.

Part II is dedicated to the world of wavelets and comprises an introduction (Chapter 5), the description of a wavelet based spatial audio format (Chapter 6), and finally our special wavelet transform (Chapter 7).

Part III is dedicated to the evaluation of the new format.

This thesis sets a new approach to spatial audio encoding, that bridges the channel-based approaches with the channel-agnostic ones, widening the spectra of existing spatial audio methods, possibly generalizing and incorporating

| Technology | Encoding | Transmission | Decoding |
|---|---|---|---|
| Ambisonics | ✗ | ✗ | ✔ |
| Wavelet Framework | ✔ | ✔ | ✔ |

Table 1.1: Table of contributions.

already existing theories.

## 1.4 Original Contributions

Novel contributions produced in the context of this thesis are: Chapter 3 description of a generic method for decoding of linear-encoding formats to irregular layouts, the evaluation of this approach in Chapter 4, the new wavelet based spherical audio framework described in Chapter 6, the optimization of the wavelet filters for spatial audio purposes in Chapter 7, and the whole evaluation, Part III, is an original contribution as well.

Table 1.1 shows graphically the contributions to the Ambisonics and Wavelet format audio chains, that are the result of this work.

## 1.5 Outline

After a first introductory Chapter 1, the thesis est omnis divisa in partes tres: Part I on Ambisonics, Part II on wavelets and the wavelet spatial audio framework we designed, and in the last Part III an incarnation of this framework into a spherical audio format is evaluated against Ambisonics.

Part I is composed of three Chapters, the first summarizes the background information and the following ones describe the original contributions. Chapter 2 briefly describes Ambisonics giving the basis for encoding and decoding Higher Order Ambisonics. Chapter 3 is an original contribution to Ambisonics' decoding to irregular speakers' layouts. In the first Section we describe the physical and psychoacoustical variables to design the Ambisonics decoder. We formulate the problem as an optimization problem, so in the subsequent Sec-

tion we define the optimization's cost function. Finally we show the resulting performance of the decoder for a specific layout of speakers. In Chapter 4 we evaluate the performance of our decoder against some publicly available ones, both objectively and subjectively.

Part II is composed of three Chapters, the first summarizes the background information and the following ones describe the original contributions. The first Chapter 5 is an introduction to Wavelet Theory, starting from a comparison with Fourier Transform and then fast-forwarding into multiresolution, the Lifting Scheme and a construction of Spherical Wavelets. Chapter 6 describes a method to generate spherical audio formats using wavelets built on a multiresolution mesh.

Chapter 7 illustrates a numerical method to obtain spherical wavelet filters optimized for spatial audio purposes.

The last Part III is composed of two Chapters, both are new contributions. Chapter 8 evaluates different versions of Spherical Audio Formats defined by different wavelet families. Chapter 9 inspects the properties of this new format against Ambisonics for a reference layout of speakers.

The material presented in Part I is an updated version of two already published contributions: a peer-reviewed conference proceeding [Scaini and Arteaga, 2014] and a conference proceeding with a peer-reviewd abstract [Scaini and Arteaga, 2015]. Some of the material presented in Part II is part of an article already submitted to the *Journal of the Audio Engineering Society*, with title "Wavelet based spherical audio format" focused on the audio workflow of the new wavelet framework [Scaini and Arteaga, 2019b]. Another article devoted to the wavelet optimization method is in preparation [Scaini and Arteaga, 2019a].

# Part I

# Ambisonics

# Chapter 2

# INTRODUCTION TO AMBISONICS

Ambisonics is a theory for spatial audio recording and reproduction, developed by Michael Gerzon during 1970s, that aims at the encoding of the sound field and its accurate reconstruction in a point in space. From a theoretical point of view, it is possible to define the Ambisonics channels as the coefficients of a perturbative series expansion in terms of spherical harmonics (SH) of the sound field around the origin.

Zeroth order Ambisonics consists of one channel, the $W$ channel, is the omnidirectional component of the field, and corresponds to the sound pressure. First order Ambisonics (FOA) adds the $X$, $Y$ and $Z$ channels, are the directional components in three dimensions, and correspond to the three components of the pressure gradient, which amount at the acoustic velocity at the origin (see Table 2.1). Together, these components approximate, at first order of the multipole expansion, the sound field on a sphere around the listening point. $L$-th order Ambisonics adds other coefficients to the multipole expansion which amount to quantities proportional to derivatives (up to $L$-th order) of the pressure field.

Higher Order Ambisonics (HOA) is made of $K = (L+1)^2$ channels, where $L$ is the Ambisonics order or spherical harmonic degree (each $l$ order has $2l+1$ channels). In the following we will refer to an arbitrary Ambisonics

| Ch. # | (l,m) | Name | physical meaning |
|---|---|---|---|
| 0 | (0,0) | W | pressure |
| 1 | (1,-1) | Y | particle velocity $y$ dir. |
| 2 | (1,0) | Z | particle velocity $z$ dir. |
| 3 | (1,1) | X | particle velocity $x$ dir. |

Table 2.1: First Order Ambisonics Channels.

order, including HOA, simply as *Ambisonics*.

Ambisonics has a series of remarkable properties, which make it a good format for spatial audio. First, it is a complete theory of spatial audio, going from recording to reproduction. Second, it is based on solid acoustic and mathematical grounds. Third, it is has a fixed number of channels, independently from the number of sound sources (in contrast to object-based methods). Fourth, it is completely independent of the exhibition layout. Fifth, it provides a smooth listening experience from all directions. Finally, depending on the order, it requires only a moderate number of loudspeakers for exhibition (if compared e.g. with Wave Field Synthesis).

## 2.1 Encoding Higher Order Ambisonics

The decomposition of a distribution of sources $S(\theta, \phi)$ over a sphere is expressed, in $L$-th order Ambisonics,

$$S(\theta, \phi; t) = \sum_{l=0}^{L} \sum_{m=-l}^{l} a_{l,m}(t) Y_{l,m}(\theta, \phi) \tag{2.1}$$

where $Y_{l,m}(\theta, \phi)$ are the real-valued spherical harmonics [1] (which form a basis in $S^2$) [Arfken et al., 2005], and $a_{l,m}$ are the coefficients or the projection of

[1] Following the convention in `http://ambisonics.ch/standards/channels/glossary`.

$S$ onto the basis $Y$:

$$\begin{aligned} a_{l,m}(t) &= \int S(\theta,\phi;t)Y_{l,m}(\theta,\phi)\mathrm{d}\Omega \\ &= \int_0^{2\pi}\int_{-\pi/2}^{\pi/2} S(\theta,\phi;t)Y_{l,m}(\theta,\phi)\cos(\theta)\mathrm{d}\phi\mathrm{d}\theta \end{aligned}$$

where $\mathrm{d}\Omega$ is the usual measure on the sphere, $\mathrm{d}\Omega = \cos\phi\mathrm{d}\phi\mathrm{d}\theta$ in acoustic coordinates convention $\phi$ is azimuth and $\theta$ is elevation. The distribution of sources can be linked to the perturbative decomposition of the pressure field around the origin [Daniel et al., 2003]. The coefficients $a_{l,m}(t)$ are the so-called *Ambisonics channels*.

A plane wave $S_{\hat{\mathbf{u}}}(\theta,\phi)$, representing a virtual point source coming from direction [2] $\hat{\mathbf{u}}(\theta_{\hat{\mathbf{u}}},\phi_{\hat{\mathbf{u}}})$, can be approximated in terms of spherical harmonics as:

$$S_{\hat{\mathbf{u}}}(\theta,\phi) = \sum_{l=0}^{L}\sum_{m=-l}^{l}(a_{\hat{\mathbf{u}}})_{l,m}Y_{l,m}(\theta,\phi), \tag{2.2}$$

where the Ambisonics coefficients $(a_{\hat{\mathbf{u}}})_{l,m}$ are given in the normalization N3D [Daniel, 2000] by:

$$(a_{\hat{\mathbf{u}}})_{l,m} = g_{\hat{\mathbf{u}}}Y_{l,m}(\theta_{\hat{\mathbf{u}}},\phi_{\hat{\mathbf{u}}}), \tag{2.3}$$

where $g_{\hat{\mathbf{u}}}$ is the amplitude of the plane wave. The importance of plane waves lies in the fact that any distribution of sources can be represented as a superposition of plane waves.

These expressions can be rewritten in matrix notation. Equation (2.1) can be reexpressed as:

$$S(\theta,\phi) = \mathbf{a}\cdot\mathbf{Y}(\theta,\phi)$$

with

$$\begin{aligned} \mathbf{a} = (&a_{0,0}\ \ldots a_{1,-1}\ \ldots\ a_{1,0}\ \ldots\ a_{1,1}\ldots \\ &\underbrace{a_{l,-l}\ \ldots\ a_{l,0}\ \ldots\ a_{l,l}}_{2l+1}\ \ldots\ a_{L,-L}\ \ldots\ a_{L,0}\ \ldots\ a_{L,L})^T \end{aligned}$$

[2] Hat denotes unit vectors in $S^2$.

and

$$\mathbf{Y} = (Y_{0,0} \ ... \ Y_{1,-1} \ ... \ Y_{1,0} \ ... \ Y_{1,1} \ ... \\ Y_{l,-l} \ ... \ Y_{l,0} \ ... \ Y_{l,l} \ ... \ Y_{L,-L} \ ... \ Y_{L,0} \ ... \ Y_{L,L})^T.$$

The encoding equation (2.3) for a point source can be reexpressed as:

$$\mathbf{a}_{\hat{\mathbf{u}}} = g_{\hat{\mathbf{u}}} \mathbf{Y}(\theta_{\hat{\mathbf{u}}}, \phi_{\hat{\mathbf{u}}})$$

Explicitly, with the choice of the N3D convention mentioned above:

$$\begin{cases} W_{\hat{\mathbf{u}}} = (a_{\hat{\mathbf{u}}})_{0,0} & = S \\ Y_{\hat{\mathbf{u}}} = (a_{\hat{\mathbf{u}}})_{1,-1} & = S\sqrt{3}\cos(\theta_{\hat{\mathbf{u}}})\sin(\phi_{\hat{\mathbf{u}}}) \\ Z_{\hat{\mathbf{u}}} = (a_{\hat{\mathbf{u}}})_{1,0} & = S\sqrt{3}\sin(\theta_{\hat{\mathbf{u}}}) \\ X_{\hat{\mathbf{u}}} = (a_{\hat{\mathbf{u}}})_{1,1} & = S\sqrt{3}\cos(\theta_{\hat{\mathbf{u}}})\cos(\phi_{\hat{\mathbf{u}}}) \\ ... \end{cases}$$

## 2.2 Decoding Higher Order Ambisonics

### 2.2.1 Basic Ambisonics Decoding

The *basic decoding* assumes phase coherence among signals emitted by the loudspeakers, and the requirement is to accurately reconstruct the sound field at the origin up to order $L$ in the Ambisonics decoding, from superposition of the plane waves emitted by the different loudspeakers in the layout[3]. In this case the problem is linear and can be solved analytically with algebraic methods.

With more detail, let us define a set of directions $\Theta = \{\hat{\mathbf{u}}_i\}_{i=1,...,N}$, where $\hat{\mathbf{u}}_i(\theta_i, \phi_i) \in S^2$ correspond to the position of the loudspeakers, which sample the $K = (L+1)^2$ spherical harmonics up to $L^{\text{th}}$ degree:

$$\mathbf{Y} = (Y_{l_1,m_1} \ ... \ Y_{l_L,m_L})^T,$$

[3]Note that it is assumed that the loudspeakers emit plane waves and are placed at the same distance. Compensation of level, delay and near-field effect [Daniel, 2003] has to be addressed in another stage of the signal processing.

in $N$ directions:

$$\mathbf{y}_{l,m} = (Y_{l,m}(\theta_1, \phi_1) \ \dots \ Y_{l,m}(\theta_N, \phi_N))^T.$$

The decoding equation requests that the original sound field represented by Eq. (2.1) is accurately reproduced up to order $L$ from the plane waves emitted from the different $\{\hat{\mathbf{u}}_i\}_{i=1,\dots,N}$ loudspeakers' directions, given by (2.2):

$$\mathbf{a} = \sum_{j=1}^{N} g_j \mathbf{Y}(\theta_j, \phi_j), \tag{2.4}$$

where $g_i$, the gain of each one of the loudspeakers, is the unknown in the above equation. At first Ambisonics order this request amounts to reproducing correctly the first four spherical harmonics, corresponding to the sound pressure $p$ and normalized acoustic velocity at the origin $\mathbf{v}$. Eq. (2.4) can be reexpressed in matrix form more concisely as:

$$\mathbf{a} = \mathbf{C}\,\mathbf{g},$$

where the matrix $\mathbf{C} = \{c_{k,j}\}$ represents the sampled spherical harmonics basis

$$\mathbf{C} = \begin{pmatrix} Y_{0,0}(\theta_1, \phi_1) & \cdots & Y_{0,0}(\theta_N, \phi_N) \\ Y_{1,-1}(\theta_1, \phi_1) & \cdots & Y_{1,-1}(\theta_N, \phi_N) \\ Y_{1,0}(\theta_1, \phi_1) & \cdots & Y_{1,0}(\theta_N, \phi_N) \\ Y_{1,1}(\theta_1, \phi_1) & \cdots & Y_{1,1}(\theta_N, \phi_N) \\ \vdots & c_{k,j} & \vdots \\ Y_{L,-L}(\theta_1, \phi_1) & \cdots & Y_{L,-L}(\theta_N, \phi_N) \\ \vdots & \vdots & \vdots \\ Y_{L,0}(\theta_1, \phi_1) & \cdots & Y_{L,0}(\theta_N, \phi_N) \\ \vdots & \vdots & \vdots \\ Y_{L,L}(\theta_1, \phi_1) & \cdots & Y_{L,L}(\theta_N, \phi_N) \end{pmatrix}_{\substack{k=1,\dots,K; \\ i=1,\dots,N}}$$

and $\mathbf{g} = (g_1 \dots g_N)^T$ is the vector of gains.

The decoding matrix $\mathbf{D}$ is the matrix giving:

$$\mathbf{g} = \mathbf{D}\,\mathbf{a}.$$

In most realistic cases the system is under-determined, given that the number of loudspeakers $N$ is generally greater than the number of Ambisonics channels $K$. From all the possible solutions to the system, the pseudoinverse is the one that minimizes the energy emitted [Daniel, 2000]:

$$\mathbf{D} = \mathbf{D}_{\text{pinv}} = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}. \tag{2.5}$$

Eq. (2.5) represents the general solution for the basic decoding. However, it is to be noted that in highly irregular layouts the inverse in Eq. (2.5) is ill-conditioned and a regularization should be applied [Zotter et al., 2012].

The set of sampling directions $\Theta$ is said to be *regular* for the basis $\mathbf{Y}$ if it preserves the orthonormality of the sampled basis [Daniel, 2000]. This means that $\mathbf{C}\mathbf{C}^T/N = \mathbf{1}_K$, where $\mathbf{1}_K$ is the unity matrix of range $K$. The set of sampling directions $\Theta$ is said to be *semi-regular* for the basis $\mathbf{Y}$ if it preserves the orthogonality of the sampled basis. This means that $\mathbf{C}\mathbf{C}^T$ is diagonal. If the set of sampling directions $\Theta$ does not preserve any of the previous properties, then it is said to be *irregular*[4]. As a further clarification, in Ambisonics terms a regular set of directions does not necessarily imply a 'regularly spaced' set of directions.

If the set of sampling directions is regular, then then the decoding matrix becomes:

$$\mathbf{D} = \mathbf{D}_{\text{proj}} = \mathbf{C}^T/N. \tag{2.6}$$

Namely, the decoding equations consist on a mere projection of the spherical harmonics on the corresponding loudspeaker direction. The decoding matrix obtained via projection is often referred to as 'naïve decoding'.

### 2.2.2 Modified Psychoacoustical Decodings

Psychoacoustically, the basic decoding method is optimal at low frequencies, below 500 Hz approximately, and close to the sweet spot. At higher frequencies or for a large listening area it is preferable to use modified psychoacoustic decodings [Daniel, 2000].

[4] An alternative definition, makes use of spherical $t$-designs, with $t \geq 2L + 1$, which identifies the optimal loudspeaker arrangement [Zotter and Frank, 2012].

Let $\{\hat{\mathbf{v}}_j\}$ be a set of $n$ directions sampling the sphere (useful later). Given a decoding $\mathbf{D}$, the signal fed to the speaker $i$ while reproducing a virtual plane wave of unit amplitude coming from direction $j$ will be labelled $s_{ij}$ (it is actually given by $s_{ij} = (\mathbf{D}\,\mathbf{Y}(\theta_j, \phi_j))_i$).

The *max-$r_E$ decoding* assumes incoherent sum of the speaker signals. For regular layouts, the modified decodings can be computed by requiring that the decoding reproduces the original energy and acoustic intensity at the origin. Within the incoherent sum hypothesis, and assuming that each one of the incoming waves is a plane wave, a statistical estimator of the signal energy $E_j$ at the origin is:

$$E_j = \sum_{i=1}^{n} |s_{ij}|^2$$

and a statistical estimator of the normalised acoustic intensity $\mathbf{I}_j$ is

$$\mathbf{I}_j = \frac{1}{E_j} \sum_{i=1}^{n} |s_{ij}|^2 \hat{\mathbf{u}}_i = r_E \hat{\mathbf{v}}_E.$$

This decoding requests:

$$E_j = 1,$$
$$\mathbf{I}_j = \hat{\mathbf{v}}_j \implies \begin{cases} r_E = 1, \\ \hat{\mathbf{v}}_E = \hat{\mathbf{v}}_j \end{cases}$$

It is physically impossible to fulfill the condition $r_E = 1$ by summing incoherently the signal of several loudspeakers; decodings will instead try to maximise this value (hence the name). Psychoacoustically, this decoding reproduces the impression of the original sound at high frequencies, above 500 Hz approximately.

The *in-phase* decoding imposes the additional restriction that there are no loudspeakers emitting in opposite phase. This decoding gives a more robust localisation for listeners who are far from the sweet spot.

While there is experimental evidence that $\mathbf{I}_j$ gives a good indicator of the perceived source direction, and that a frequency-weighted version of $r_E$ is a

good indicator of the perceived source width for broadband signals [Frank, 2013], a detailed analysis of the optimal localization criteria depending on the frequency is out of the scope of this thesis. However, let us stress the key feature of the max-$r_E$ or inphase decodings is the incoherent summation hypothesis rather than the specific localization criteria.

For regular or semi-regular layouts, the intensity vector under the incoherent sum hypothesis is parallel to the acoustic velocity vector in the coherent sum hypothesis and it is possible to obtain the optimal max-$r_E$ or in-phase decodings by doing slight modifications of the regular decoding, Eq. (2.6) [Daniel, 2000]. However for non-regular layouts the velocity and intensity vectors in the two hypothesis are not parallel, the problem is fully nonlinear and algebraic methods are not helpful. To solve this case a nonlinear method can be used.

The decoding equations have closed analytic expressions only for regular loudspeaker arrays [Daniel, 2000], but speaker layouts in real-world installations are often irregular (in the Ambisonics sense).

# Chapter 3

# AMBISONICS DECODING TO IRREGULAR LAYOUTS

Higher Order Ambisonics has some drawbacks which hinder its widespread adoption. First, the directionality properties of sound sources encoded in low order Ambisonics is often regarded as poor. Second, the sweet spot, the area where the reconstruction is optimal, is small. These two drawbacks can be ameliorated by going to higher order Ambisonics. Additionally, decoding Ambisonics to non-regular loudspeaker layouts is challenging. The decoding equations for HOA have closed analytic expressions only for regular loudspeaker arrays [Gerzon and Barton, 1992, Daniel, 2000], but most real-world layouts, like the ubiquitous stereo, 5.1 and 7.1 surround configurations, are non-regular from the Ambisonics point of view. The generation of optimal and psychoacoustically correct decodings for irregular loudspeakers layouts is a nonlinear problem which can be solved using numerical search algorithms. In this section we address this problem by presenting an algorithm for higher order Ambisonics decoding, together with its open source implementation, IDHOA [Scaini, 2015].

There are several previous references in the literature calculating the decoding of Ambisonics for irregular loudspeaker arrays. The papers [Wiggins et al., 2003, Wiggins, 2004] and [Moore and Wakefield, 2007, Moore and Wakefield, 2011] concentrate on decoding algorithm for the 5.1 ITU layout based on a

modified tabu search algorithm. Tsang et al. similarly have a decoding strategy based on genetic algorithms and neural networks [Tsang and Cheung, 2009] while [Benjamin et al., 2010, Heller et al., 2012] work with a preexisting nonlinear optimisation library.

The algorithm presented here follows the method of [Arteaga, 2013], but extends the technique up to 5th order Ambisonics. The method differs from some of the above references in the decoding technique employed, the fitting functions employed and the focus on 3D layouts (see Section 3.4 for further details). Besides the methodology, the implementation is very different.

It is to be noted that there are also alternative decoding techniques which do not involve nonlinear search methods, like decoding to an intermediate regular layout, and later on using VBAP for decoding [Batke and Keiler, 2010, Boehm, 2011] (see however the comments in [Schmele et al., 2013]). Or, again, modifying the basic Ambisonics decoding to ensure preservation of the energy with Energy Preserving Ambisonics Decoder (EPAD) [Zotter et al., 2012] and All-Round Ambisonic Decoding (AllRAD) [Zotter and Frank, 2012, Zotter and Frank, 2018]. In [Zotter and Frank, 2019] the authors make a good comparison between different Ambisonics decoding techniques, with particular focus on the projection method, mode matching decoder (MAD) [Poletti, 2005], the EPAD and All-RAD decoders.

The results presented in this Chaper are based on the paper [Scaini and Arteaga, 2014].

## 3.1 Psychoacoustically motivated numerical optimization of an Ambisonics decoder

The decoding matrix is computed minimising a function by means of a multidimensional search algorithm. In this section the physical variables ($E$, $I^R$, $I^T$) used to calculate the function to be minimized and the objective function itself are defined.

Assuming $n$ different directions sampling the sphere, the energy and

| Order | max-$r_E$ | in-phase |
|---|---|---|
| 1 | 0.577 | 0.500 |
| 2 | 0.775 | 0.667 |
| 3 | 0.861 | 0.750 |
| 4 | 0.906 | 0.800 |
| 5 | 0.932 | 0.833 |

Table 3.1: Maximum theoretical values for $I_j^R$ (or $r_E$) for a regular layout for max-$r_E$ and in-phase decodings.

acoustic intensity generated at the origin are:

$$E_j = \sum_{i=1}^{n} |s_{ij}|^2$$

$$\mathbf{I}_j = \frac{1}{E_j} \sum_{i=1}^{n} |s_{ij}|^2 \hat{\mathbf{u}}_i,$$

where $s_{ij}$ is the signal emitted by the loudspeaker $i$, when reproducing a sound source coming from direction $j$.

The vector $\mathbf{I}_j$ can be projected in the radial and transverse part as follows:

$$I_j^R = \mathbf{I} \cdot \hat{\mathbf{v}}_j = \frac{1}{E_j} \sum_{i=1}^{n} |s_{ij}|^2 \, \hat{\mathbf{u}}_i \cdot \hat{\mathbf{v}}_j,$$

$$I_j^T = ||\mathbf{I} \times \hat{\mathbf{v}}_j|| = \frac{1}{E_j} \sum_{i=1}^{n} |s_{ij}|^2 \, ||\hat{\mathbf{u}}_i \times \hat{\mathbf{v}}_j||.$$

The radial part $I_j^R$ represents the desired component of the intensity vector, and the tangential part $I_j^T$ represents the unwanted component. In an ideal decoding, $E_j = 1$, $I_j^R = 1$ and $I_j^T = 0$, but regular (ideal) layouts the values for $I_j^R$ are always $I_j^R < 1$ (see Table 3.1).

Note that the decomposition in Eq. (3.1) is different from the decomposition used in several previous irregular decoding references [Wiggins et al.,

2003, Wiggins, 2004, Moore and Wakefield, 2007, Moore and Wakefield, 2011]: instead of maximising the norm of the intensity vector, and minimising the angle mismatch, it proves more natural and effective to maximise the radial vector components and minimise the tangential components.

From this quantities different cost function terms can be defined:

$$C_E = \frac{1}{n}\sum_{j=1}^{n}(1 - E_j)^2 w_j, \tag{3.2}$$

$$C_{\mathrm{IR}} = \frac{1}{n}\sum_{j=1}^{n}(1 - I_j^R)^2 w_j, \tag{3.3}$$

$$C_{\mathrm{IT}} = \frac{1}{n}\sum_{j=1}^{n}(I_j^T)^2 w_j, \tag{3.4}$$

where $w_j$ is possible weighting function (see Section 3.2.2 for more details). These contributions can be interpreted as follows: $C_E$ is the mean quadratic deviation from the correct level normalisation; $C_{\mathrm{IR}}$ is the mean quadratic deviation from the optimal directionality; $C_{\mathrm{IR}} > 0$ means that the directionality of the sources is not optimal, and, finally, $C_{\mathrm{IT}}$ is the mean quadratic value of the wrong component of the direction;

In the case of the in-phase decoding, there is an extra cost function to take into account:

$$E_j^{\mathrm{ph}} = \sum_{i=1}^{n} |s_{ij}|^2 \theta(-s_{ij}),$$

$$C_{\mathrm{ph}} = \frac{1}{n}\sum_{j=1}^{n}(E_j^{\mathrm{ph}})^2 w_j,$$

where $\theta(\cdot)$ is the Heaviside step function.

While physically it is possible to obtain $C_E = C_{\mathrm{IT}} = 0$, it is impossible to get the ideal decoding with $C_{\mathrm{IR}} = 0$, i.e. $I^R = 1$.

Similarly to the energy and intensity terms, it is possible to define some cost function terms for the pressure and velocity (radial and transverse), which are

the relevant quantities for the low frequencies decoders. Being

$$P_j = \sum_{i=1}^{n} |s_{ij}|^2$$

$$\mathbf{V}_j = \sum_{i=1}^{n} s_{ij}\hat{\mathbf{u}}_i,$$

the pressure $P$ and the particle velocity $\mathbf{v}$, we define the cost function terms as:

$$C_P = \frac{1}{n}\sum_{j=1}^{n}(1 - P_j)^2 w_j, \tag{3.5}$$

$$C_{\text{VR}} = \frac{1}{n}\sum_{j=1}^{n}(1 - I_j^R)^2 w_j, \tag{3.6}$$

$$C_{\text{VT}} = \frac{1}{n}\sum_{j=1}^{n}(I_j^T)^2 w_j, \tag{3.7}$$

Finally, the different cost function terms are combined to give the objective function to be minimised:

$$\begin{aligned} f = {} & \alpha_P C_P + \alpha_{\text{VR}} C_{\text{VR}} + \alpha_{\text{VT}} C_{\text{VT}} \\ & + \alpha_E C_E + \alpha_{\text{IR}} C_{\text{IR}} + \alpha_{\text{IT}} C_{\text{IT}} \\ & + \alpha_{\text{ph}} C_{\text{ph}}. \end{aligned} \tag{3.8}$$

The values of the coefficients $\alpha_P$, $\alpha_{\text{VR}}$, $\alpha_{\text{VT}}$ (basic), $\alpha_E$, $\alpha_{\text{IR}}$, $\alpha_{\text{IT}}$ (max-$r_E$) and $\alpha_{\text{ph}}$ (in-phase) can be selected at will.

## 3.2 IDHOA Decoder

### 3.2.1 The Decoder Strategy

The IDHOA decoder that calculates decoding matrices for Ambisonics up to order 5. The decoder is based on the minimization of the objective function in (3.8). Our implementation of IDHOA [Scaini, 2015] makes use of

Python [van Rossum, 1995], IPOPT [Wächter and Biegler, 2006] and PyTorch [Paszke et al., 2017]. The flow of the algorithm can be summarized as follows:

1. *Initialization:* Operations that are performed only once when the algorithm is launched.

2. Given the loudspeakers' layout, calculate $\mathbf{D}$, both Eq. (2.6) and Eq. (2.5).

3. Calculate the various physical variables that constitute the objective function, $p$, $E$, $\mathbf{v}$, $\mathbf{I}$, over the $n$ sampling directions. Calculate the objective function, which is $f = f(\mathbf{D})$.

4. Select the $\mathbf{D}$ matrix that minimizes $f$, $f = \min\{f(\mathbf{D}_{\text{proj}}), f(\mathbf{D}_{\text{pinv}})\}$, to be used as the initial point for the minimization algorithm $\mathbf{D}_{\text{init}}$.

5. *Fix constraints (optional):* constrain some parameters to have a fixed value (e.g. lock to zero).

6. *Minimization stage:* Call to the external minimization algorithm, passing $\mathbf{D}_{\text{init}}$ and $f$. When the minimization algorithm terminates, it returns a $\widetilde{\mathbf{D}}$.

### 3.2.2 Configuration of IDHOA

In the IDHOA decoder it is possible to tune several parameters to obtain the desired Ambisonics decoder. In the following we will detail the compulsory and optional ones.

**Layout** First to be provided, are the coordinates of the target layout $(\theta, \phi)$. It is also possible to provide cartesian $(x, y, z)$ coordinates. Cartesian coordinates will be converted to $(\theta, \phi)$, stripping out the distance $R$ information. The hypothesis is that the distance will be addressed in another stage of the decoding, with proper delays and near-field filters.

**Basic settings** *Degree.* The decoder allows to generate the Ambisonics decoding coefficients up to fifth order, setting the `DEG` variable to the desired order. *Decoding scheme.* In the first implementations of IDHOA described in published papers, the `DEC` variable allowed to choose between the three different decodings: basic, max-r$_E$, in-phase. In the recent years we decided to simplify this approach and have only a *universal* decoding, where the different requests for the optimization are balanced with the $\alpha_E$, $\alpha_P$, ... , coefficients directly. This choice has two motivations: firstly, in HOA it is possible to reconstruct pressure and energy at the same time[1], so separating the decoder in two (or three) completely different decoders can be avoided, or the transition between the different decors can be made smoother. Secondly, the rigid separation in different cost functions depending of the decoding scheme was a limitation during the experimentation with wavelet format decoding described in Part II.

**Optional parameters** The weighting function $w_j$ in the definition of the various terms (Eqs. (3.2) and following) is an optional biasing factor which allows to improve the decoding performance in some regions of the sphere (at the expense of other regions). A non-biased decoding is given by $w_j = 1$. Some examples of possible weighting functions are reported in [Arteaga, 2013]. For example, it is possible to use a function that masks automatically the areas with no loudspeakers; being $\hat{\mathbf{u}}$ the direction of the loudspeaker, and $\hat{\mathbf{v}}$ the direction where the function is being evaluated:

$$w_j = \begin{cases} 1 & \text{if } d(\hat{\mathbf{u}}, \hat{\mathbf{v}}) < \tilde{d}, \\ \beta & \text{if } d(\hat{\mathbf{u}}, \hat{\mathbf{v}}) \geq \tilde{d}, \end{cases}$$

[1] It is possible also at FOA, but the price to pay to have a linear (pressure) and a quadratic (energy) quantity (which are function of the speakers' gains) that sum to 1, is having negative gains. This is typically a bad idea, because negative gains mean out-of-phase speakers that results in a limited (depending on frequency) area where the signals sum properly, i.e. small sweet spot.

where $\beta$ is a parameter that can be tuned[2] with $0 \leq \beta \leq 1$, $d$ is the angular distance calculated with the Haversine formula [Bureau, 1997], and $\tilde{d}$ is a fixed threshold (a reasonable value can be 1.5 times the mean great-circle distance between the loudspeakers). This approach is quite flexible and can be modified to fit other preferences.

Another interesting feature is the preservation of the natural left/right symmetry of the speakers layout in the generated decoding matrix. One part of the algorithm searches for left-right speakers and pairs them, reducing the effective number of degrees of freedom, and fixing this symmetry into the optimized decoding matrix **D**.

In our Python implementation, these parameters are set in the `layout_name.ini` file.

## 3.3 Performance of the Decoder

The IDHOA decoder receives as input the speakers' layout and builds the objective function to be minimised calculating some physical variables of interest. It has been chosen to maximise (to one) the radial acoustic intensity and the energy and to minimise (to zero) the tangential acoustic intensity, see Eqs. (3.2)–(3.7). It is possible to generate the decoding coefficients for Ambisonics up to fifth order, with a complete freedom in the choice of the decoding scheme: it can be basic[3], max-$r_E$, in-phase or a blend of all of them.

The tests reported here were carried out on the layout of the 3D audio studio in Barcelona Media, equipped with 23 loudspeakers, placed in an irregular hemispherical configuration[4].

The results reported in this Section make reference to the version of IDHOA originally published here [Scaini and Arteaga, 2014]. The Table 3.2 shows the values of the objective function $f$ for max-$r_E$ and in-phase decod-

[2] With $\beta = 0$ the weighting behaves like a binary mask, with $\beta > 0$ the process of masking is smoother, keeping partially into account the areas without speakers.

[3] This option is given by completeness, since the pseudoinverse method renders exact results in this case

[4] The layout is explicitly given as an example in IDHOA code [Scaini, 2013].

| Decoding | $f$ Naive | $f$ Opt. | time | $n_{active}$ |
|---|---|---|---|---|
| $1^{st}$ max-$r_E$ | 74.9 | 24.36 | 235 $s$ | 12 (/23) |
| $1^{st}$ in-phase | 66.4 | 24.36 | 286 $s$ | 13 (/23) |
| $3^{rd}$ max-$r_E$ | 193.2 | 13.94 | 736 $s$ | 21 (/23) |
| $3^{rd}$ in-phase | 115.5 | 13.36 | 814 $s$ | 20 (/23) |

Table 3.2: Objective function $f$ value for different decodings at first and third order. Moreover it is reported the lasted time for the algorithm to reach the minimum, and the number of active speakers at the end of the evaluation (out of 23).

ings at first and third order, for the *naive* and optimized *decodings*. The naïve refers to the decoding equation (2.6), corrected for the desired modified decoding (max-$r_E$ or in-phase). From this table it is possible to extract a qualitative fact: the value of the objective function decreases during the optimization process. The quantity $n_{active}$ is the number of speakers left active in the decoding. Originally the "muting" of speakers and/or Ambisonics channels was obtained by running the minimization several times, and each time locking to zero the coefficients under a certain threshold. This procedure was necessary since the used minimization algorithm, `Sbplx` (reimplementation of `Subplex` [Rowan, 1990]) from `NLopt` library [Johnson, 2007], is a local derivative-free algorithm. In the new implementation, the minimization algorithm uses jacobian and hessian and gets to the similar results in one run. The change in the algorithm impacted also the execution times reported in the Table, reducing them by a factor 10 approximately.

Comparing Figures 3.1(a) and 3.1(b) it is possible to note that the energy is properly reconstructed by IDHOA at all three Ambisonics orders considered here. Looking at Figures 3.1(c), 3.1(d), 3.2(c) and 3.2(d) it is possible to highlight the effect of increasing the Ambisonics order: the radial intensity improves at higher orders, getting close to 0.8 in all directions covered by loudspeakers already for second order Ambisonics. Figures 3.1 and 3.2 are all obtained with the last version of IDHOA code, with a combination of coefficients for the cost function that mixes all the three decoding schemes.

The decoder was tested carrying out some informal listening tests, where the subjects involved noted an improved localization with HOA with respect

to FOA. Furthermore, the subjects reported Ambisonics to display smoother pannings than VBAP does. A quantitave subjective test is reported in Chapter 4 for 2D 5.0 arrays.

## 3.4 Summary

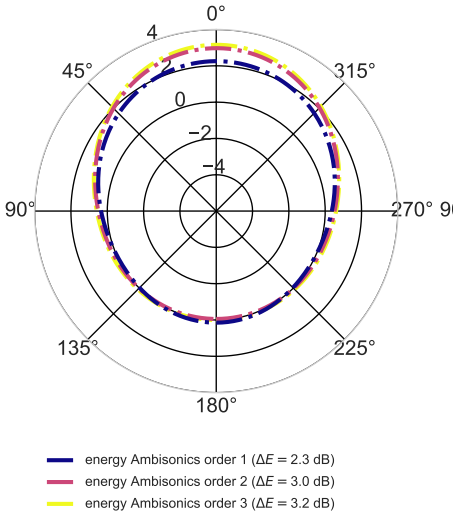
The described IDHOA decoder has been released as open-source code under the GPLv3 license, and can be downloaded at [Scaini, 2015]. The code generates a set of decoding coefficients for each loudspeaker allowing to decode Ambisonics signals up to fifth order. The strategy adopted for the search of the decoder aims to maximise the directionality of the decoded sounds on a number of sampled directions over the sphere, minimise the directional mismatch and ensure the correct sound level.

Some remarkable properties of the decoder are:

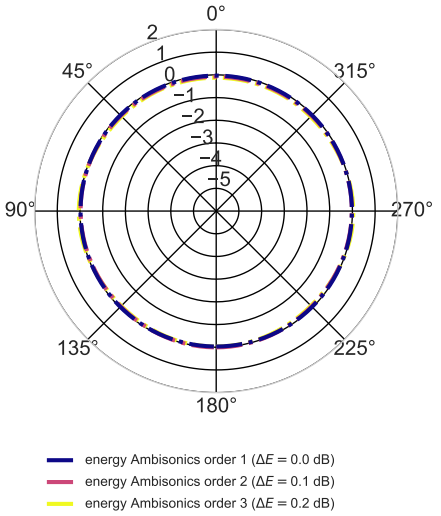
- IDHOA can generate basic, max-$r_E$, in-phase (and any almost continuous combinations of them) *periphonic* decoders up to *fifth order* of Ambisonics.
- Automatic disconnection of loudspeakers[5], and/or Ambisonics order muting[6].
- Automatic recognition of exact or approximate left/right symmetry in the layout.
- Optional weighting of some sectors of the space, to avoid trying to optimize large sectors with no speakers.
- Optional horizontal plane and frontal area weighting, to provide a better imaging in the frontal area and/or the horizontal plane.

[5] When there are more loudspeakers that the minimal number for a given order in Ambisonics, often the best decoding strategy is to use a subset of all loudspeakers.

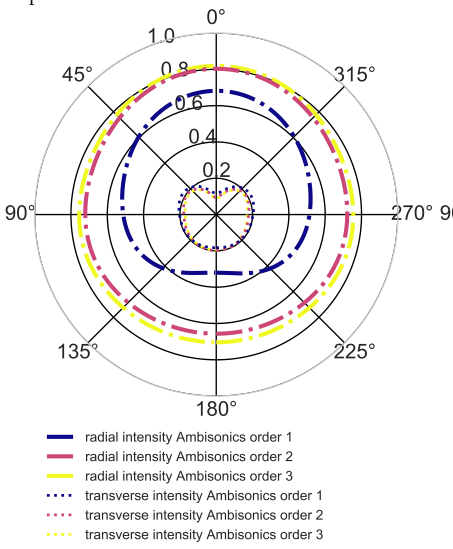
[6] It is possible for IDHOA to attempt to decode a given Ambisonics order with less speakers than channels. The orders that can't be properly decoded are automatically muted.



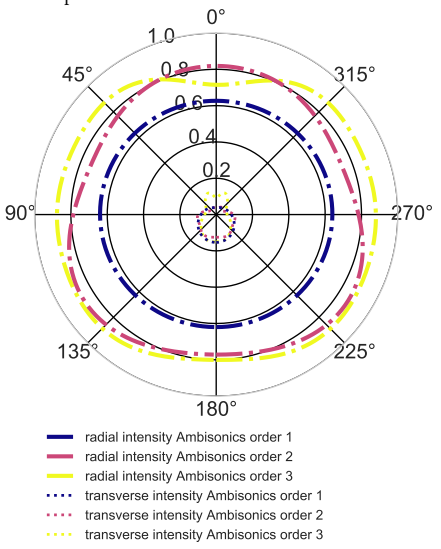

(a) reconstructed energy for naïve decoding, horizontal plane.




(b) reconstructed energy for optimized decoding, horizontal plane.

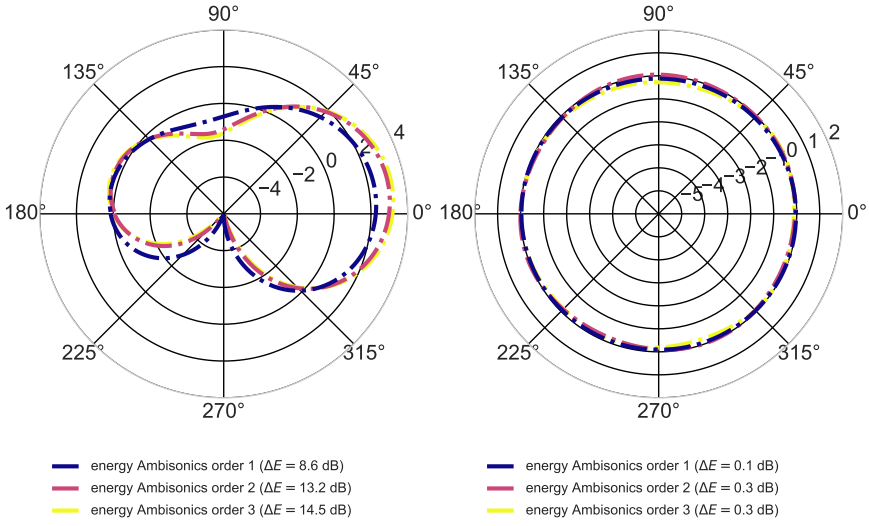



(c) reconstructed intensity for naïve decoding, horizontal plane.



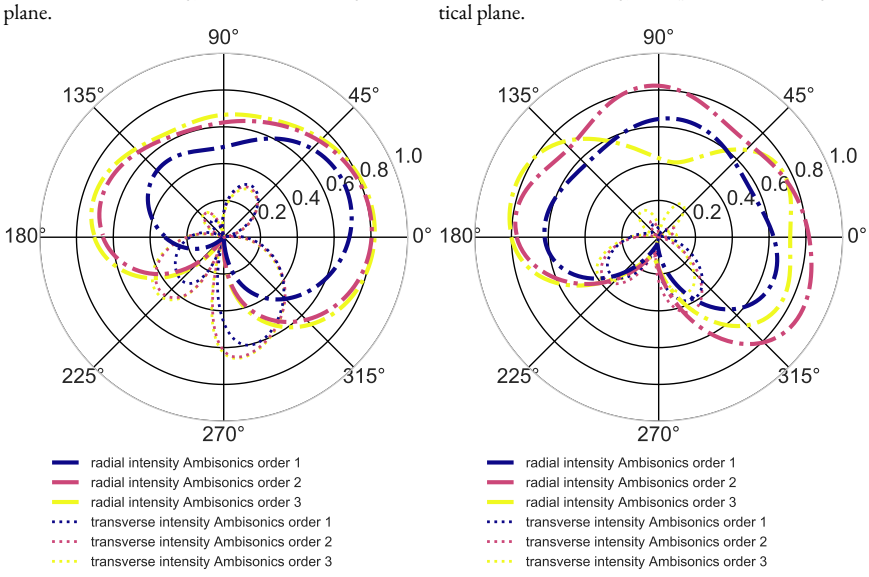

(d) reconstructed intensity for optimized decoding, horizontal plane.

Figure 3.1: First, second and third order Ambisonics. Comparison between naïve and optimized decodings for a panning around the horizontal plane, (front 0° – left 90°). The figures on the left side show the naïve decoding, while those on the right side show the optimized decodings.




(a) reconstructed energy for naïve decoding, vertical plane.

(b) reconstructed energy for optimized decoding, vertical plane.




(c) reconstructed intensity for naïve decoding, vertical plane.

(d) reconstructed intensity for optimized decoding, vertical plane.

Figure 3.2: First, second and third order Ambisonics. Comparison between naïve and optimized decodings for a panning around the vertical plane, (front 0° – up 90°). The figures on the left side show the naïve decoding, while those on the right side show the optimized decodings.

The developed decoder successfully minimises the objective function, optimizing the intensity vector and ensuring the correct energy reproduction. Informal listening tests confirm the improvement from the naïve to the optimized decoding, and detect a clear improvement in the localisation with HOA compared to FOA.

# Chapter 4

# IDHOA EVALUATION

In Chapter 3 we presented an algorithm for decoding higher order Ambisonics for irregular real-world 3D loudspeaker arrays, implemented in the form of IDHOA, an open source project. IDHOA has many features tailored for the reproduction of Ambisonics in real audio venues. In order to benchmark the performance of the decoder against other decoding solutions, we restrict the decoder to 2D layouts, and in particular to the well studied stereo, 5.1 and 7.1 surrounds.

We report on the results of the objective evaluation of the IDHOA decoder in these layouts, and of the subjective evaluation in 5.1 by benchmarking IDHOA against different decoding solutions.

This Chapter is based on the paper [Scaini and Arteaga, 2015].

## 4.1 Objective Evaluation

We generated dual-band decodings (basic decoding strategy for low frequencies, and max-$r_{\mathrm{E}}$ for high frequencies) for the stereo, 5.1 and 7.1 surround layouts, at different Ambisonics orders.

For each decoding tested, a point source has been encoded in different directions around the circle, and the following has been considered:

1. The sound level generated from each direction (values of $E$ and $p$).

| | Mean LF | | Mean HF | |
|---|---|---|---|---|
| Decoding | $v_R$ | $v_T$ | $I_R$ | $I_T$ |
| 5.0 BHL1 | 1.00 | 0.00 | 0.69 | 0.10 |
| 5.0 idhoa1 | 1.00 | 0.00 | 0.69 | 0.15 |
| 5.0 FAA2 | 1.00 | 0.00 | 0.65 | 0.01 |
| 5.0 idhoa2 | 1.00 | 0.02 | 0.78 | 0.13 |
| 5.0 idhoa3 | 1.00 | 0.02 | **0.80** | 0.14 |
| 7.0 idhoa3 | 1.01 | 0.01 | 0.87 | 0.06 |

Table 4.1: Mean values around the circle for the radial and transversal components of the velocity (basic component, low frequencies) and radial and transversal components of the intensity (max-rE component, high frequencies) for the different decodings. The best mean value for the radial intensity for the 5.0 layout is reached by the IDHOA decoding at third order, idhoa3.

2. The amount of directionality of the sound generated (values of $v_R$ and $I_R$).

3. The correctness of position of the sound source (values of $v_T$ and $I_T$).

4. The amount of crosstalk for sources panned exactly at the loudspeaker positions.

For the 5.1 layout we have additionally compared the decoders generated with select reference state-of-the-art decoders. The criteria for decoder selection was first: public availability, and second: the decoders had to target the standard ITU angular position[1].

**Layout 5.0: first order decoding** Regarding the first order decoder generated by IDHOA (from here on referred as idhoa1), the low frequency portion of the decoder reproduces correctly the pressure and velocity and is identical to the analytic decoder generate by the analytic method (pseudoinverse). The

[1] For example, the *Vienna decoders* [Gerzon and Barton, 1992] do not address the standard ITU layout.

| | Crosstalk HF (dB) | | | |
|---|---|---|---|---|
| Decoding | C | L/R | Ls/Rs | Lb/Rb |
| 2.0 idhoa1 | – | −7.2 | – | – |
| 5.0 BHL1 | **1.7** | −0.8 | −6.7 | – |
| 5.0 idhoa1 | 1.9 | 2.5 | −7.3 | – |
| 5.0 FAA2 | 1.9 | −1.8 | −4.7 | – |
| 5.0 idhoa2 | **1.7** | −4.5 | −11.3 | – |
| 5.0 idhoa3 | 2.6 | **−5.8** | **−13.0** | – |
| 7.0 idhoa3 | 2.5 | −8.1 | −3.8 | −6.8 |

Table 4.2: Total crosstalk for the different loudspeakers (crosstalk defined here as the portion of energy emitted by other loudspeakers for signals panned exactly at the loudspeaker positions, as compared to the energy emitted by that loudspeaker). The best values for the total crosstalk for the 5.0 layout are highlighted in bold.

high frequency portion of the decoder reconstructs correctly the energy and attempts to maximize the radial component of the intensity, at the expense of some localization mismatch. We activated an option in IDHOA to privilege the frontal region over the lateral and rear regions during the optimization of the decoding.

As reference decoder we have chosen the one published in [Benjamin et al., 2010], later on referred as BHL1 decoding. This decoder addresses the standard ITU 5.0 layout described before and it was obtained with a similar minimization process as in IDHOA. It is to be remarked that surprisingly the BHL1 decoder is normalized to pressure (assuming coherent addition of the low frequency signals) in the high frequency band.

As shown in Tables 4.1 and 4.2 the performance of both decoders is similar with respect to directionality properties. The main difference is the pressure normalization at high frequencies of the BHL1 decoding, which leads to a lower output level at high frequencies.

**Layout 5.0: second order decoding** The 5.0 layout has as many loudspeakers as channels in second order Ambisonics, meaning that in principle the an-

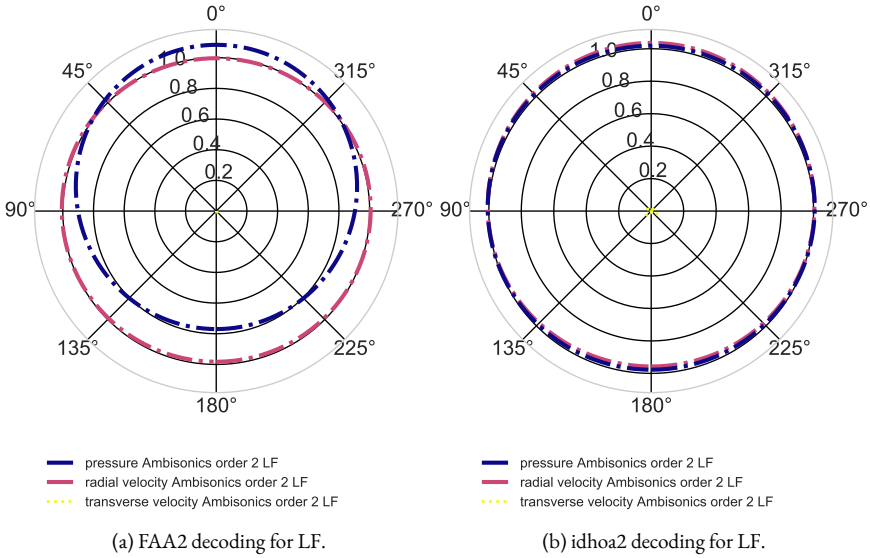
alytic inversion method could be used to reconstruct the second order Ambisonics components from the five loudspeaker signals. However, the analytic solution, while nominally correct, relies on extremely large phase cancellations which are not desired in practice. As an alternative we designed the low frequency portion of the idhoa2 decoding by requesting the correct pressure and velocity (which can already be obtained at first order), and secondarily by optimizing the intensity vector. The high frequency portion of the decoding was created using similar criteria as the first order counterpart.

The second order decoder chosen as reference is the decoder derived by Fons Adriaensen that comes with the Ambdec decoding software [Adriansen, 2015], later called simply FAA2 decoding. This decoder is one of the few public second order decoders and probably the most widespread since it is shipped with the Ambdec software.

Figure 4.1 shows the comparison between the second order decoders FAA2 and idhoa2. The FAA2 decoder (Figures 4.1(a), 4.1(c) and 4.2(a)) shows a 3 dB front dominance both at low and high frequencies. At HF the radial part of the intensity vector is maximum, 0.9, at $0^\circ$ and then decreases below 0.8 at $\pm 45^\circ$ while the localization error, represented by the tangential part of intensity, is always very small.

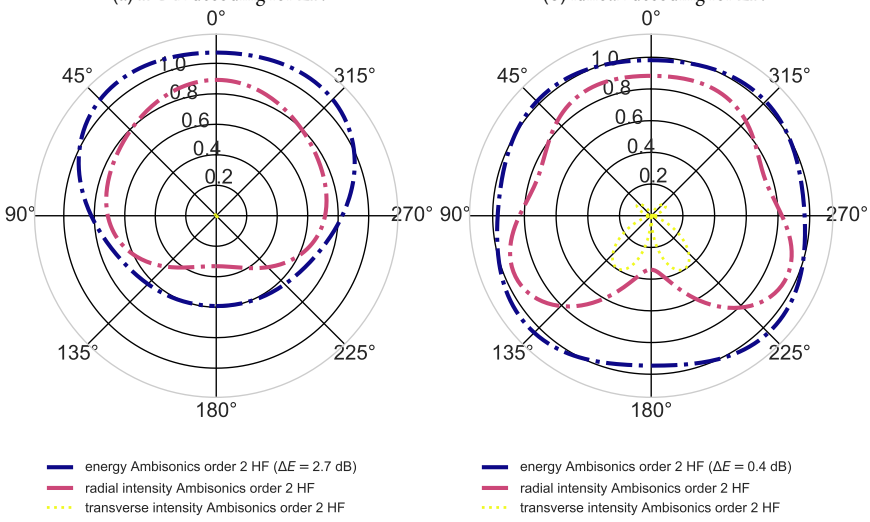
While we could possibly have generated a very similar decoder by tuning IDHOA parameters, we decided to generate a different decoder, assuring the energy to be preserved in all directions, and asking for better localization (greater radial intensity) at the expense of some directionality mismatch, see Figures 4.1(b), 4.1(d) and 4.2(b). The HF plot highlights clearly the differences between the two decoders: in idhoa2 the choice is to have good source size at the expense of some source localization mismatch where no loudspeakers are present. The reasoning behind allowing for some error in sound position is that: if the apparent source size is already "big" then a localization error is not going to be relevant. So we preferred to reduce the apparent size first and then, if the apparent source size gets sufficiently "small", optimize for its position[2]. FAA2 decoder prefers to keep small the angular error along all the circle but with an increased source size. This difference is also evident from

[2] *Optimize* an Ambisonics decoding implies always a trade off and it is the result of a deliberate (arbitrary but informed) choice.




(a) FAA2 decoding for LF.

(b) idhoa2 decoding for LF.

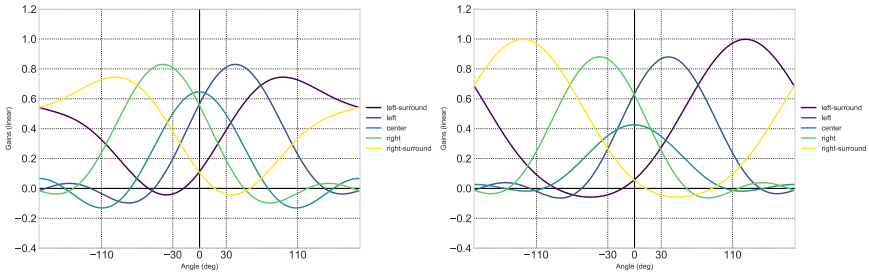



(c) FAA2 decoding for HF.




(d) idhoa2 decoding for HF.

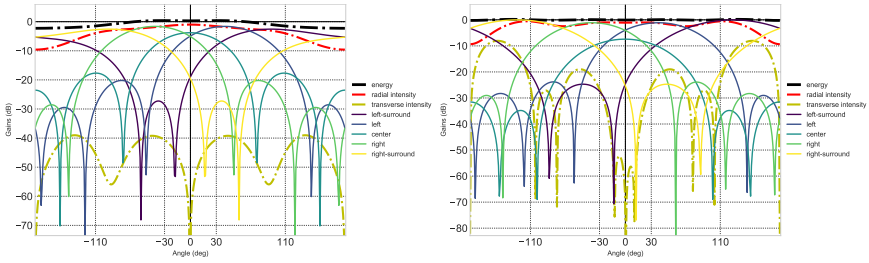
Figure 4.1: Second order Ambisonics decoders. In the left column the FAA2 decoding shipped with ambdec decoder software by Fons Adriaensen, and in the right column the one generated with IDHOA. Plots (a) and (b) show the magnitude of pressure (dotted black) and radial (dashed red) and transverse (continuous green) components of velocity as a function of the polar angle in the horizontal plane. Plot (c) and (d) show the magnitude of energy (dotted black) and radial (dashed red) and transverse (continuous green) components of intensity vector as a function of the polar angle in the horizontal plane.




(a) Gains as a function of the angle, for the HF FAA2 decoder.

(b) Gains as a function of the angle, for the HF idhoa2 decoder.

Figure 4.2: Second order Ambisonics decoders. In the left column the FAA2 decoding shipped with ambdec decoder software by Fons Adriaensen, and in the right column the one generated with IDHOA. Plots (a) and (b) show the gains of the five different loudspeakers as a function of the source position.




(a) Gains in logarithmic scale as a function of the angle, for the HF FAA2 decoder.

(b) Gains in logarithmic scale as a function of the angle, for the HF idhoa2 decoder.

Figure 4.3: Second order Ambisonics decoders. In the left column the FAA2 decoding shipped with ambdec decoder software by Fons Adriaensen, and in the right column the one generated with IDHOA. Plots (a) and (b) show the gains (in logarithmic scale) of the five different loudspeakers as a function of the source position. On the same plot is reported the reconstructed energy and intensity.

the mean values reported in Table 4.1.

Finally, let us note that the idhoa2 decoder has smaller values for the crosstalk of the lateral loudspeakers than the FAA2 decoder (see table 4.2).

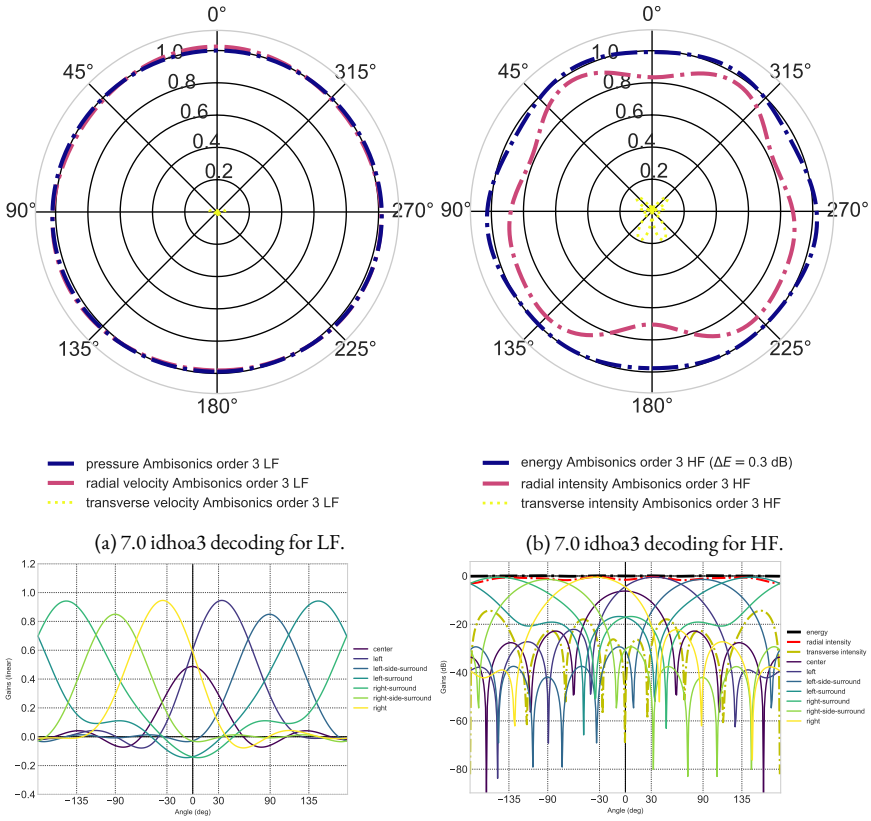
**Layout 5.0: third order decoding** In principle the third order decoding goes beyond what it can be reproduced with a 5.1 layout (there are more channels than loudspeakers). Trying to decode third order Ambisonics in a 5.1 layout can lead to some spatial aliasing, which can manifest in the form of "holes in the middle" of the loudspeaker layout.

However, IDHOA can generate a meaningful third order decoder to a 5.0 layout leading to a decoding that has better directionality properties near the loudspeaker positions, at the expense of showing the individual character of each loudspeaker. This is in contrast to traditional Ambisonics decodings, which tend to provide an approximately constant radial intensity in all directions. This way, the resulting behaviour comes closer to traditional pairwise panning.

Table 4.1 and 4.2 show that the third order decoding provides a marginally better mean directionality, with somewhat reduced crosstalk between the loudspeakers.

**Layout 7.0: third order decoding** The 7.0 layout has enough loudspeakers to decode, in principle, Ambisonics up to third order. The distribution of loudspeakers is not regular, from an Ambisonics point of view, but is indeed more homogeneous than 5.0 layout, leading to a better Ambisonics decoding. For a comparison between 5.0 and 7.0 look for example Table 4.1, the intensity components indicate better focused and localized sources in 7.0 than 5.0. Figures4.1(d) and 4.4(b) show that in 7.0 the localization properties are more uniform than in 5.0 and the minimum value for the radial intensity is larger than 0.7, which is already considered to be good.

**Layout 2.0: first order decoding** We also produced a decoder for stereo using IDHOA software, requesting an average of $-3$ dB trim in the rear part. This choice is common but arbitrary, other choices are possible and motivated by the amount of information to be mapped from the back to the front. In




(a) 7.0 idhoa3 decoding for LF.

(b) 7.0 idhoa3 decoding for HF.

(c) Gains as a function of the angle, for the HF IDHOA decoder.

(d) Gains in logarithmic scale as a function of the angle, for the HF IDHOA decoder.

Figure 4.4: Third order decoding to 7.0 layout, obtained with IDHOA software.

Figure 4.5 it is possible to see how the trim in the rear region is realized, while a good localization is achieved between $\pm 30^{\circ}$.

Anyway, in our opinion, this has to be considered as an exercise in style, since for such low number of degrees of freedom manual methods are to be preferred, given that manual fine tuning might be more predictable and adjust better to the individual preferences.

## 4.2 Subjective Evaluation

### 4.2.1 Methodology

By using a cohort of 14 subjects with age comprised between 20 and 40 years (13 males, 1 female, all of them with at least some degree of listening experience), we compared, by means of a "MUlti Stimulus test with Hidden Reference and Anchor" (MUSHRA) test [ITU-R, Recommendation BS, 2003], five different Ambisonics decoders, two state-of-the-art (first and second order), and three generated with IDHOA (first, second and third order).

The tests were performed in a treated listening room equipped with Genelec 8040B loudspeakers. The speakers' feed signals are compensated for distance, near-field effect and equalized for room coloration.

The different 5.0 decodings are assessed with respect to the following criteria:

1. The amount of directionality of the sound generated and the correctness of position of the sound source.

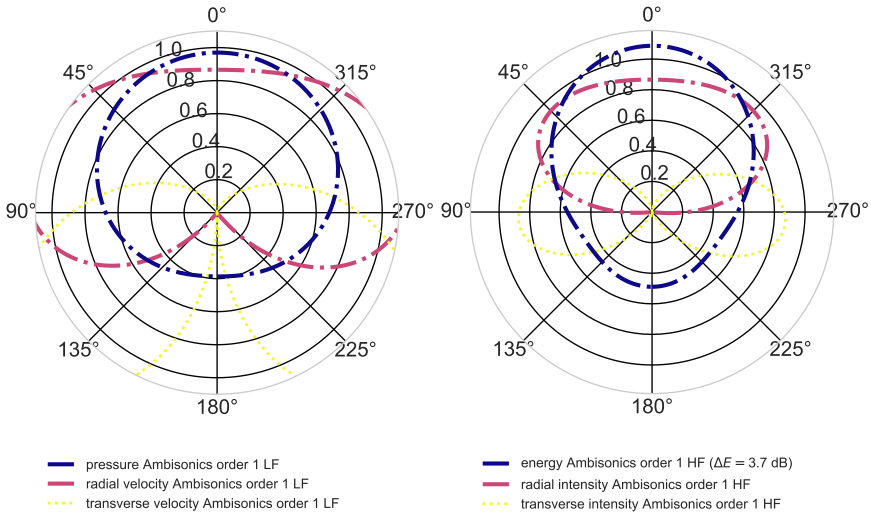
2. Smoothness of panning

3. Global spatial perception

We run three different tests:

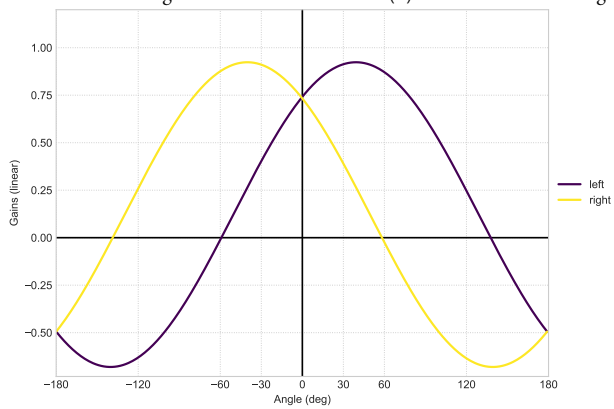
1. *Localization test.* Source positioned at $0^{\circ}$, $\pm 30^{\circ}$, $\pm 110^{\circ}$. Reference is the loudspeaker itself[3].

[3] We run some preliminary tests with a source at $90^{\circ}$ but all the decoders performed quite




(a) Stereo idhoa1 decoding for LF.

(b) Stereo idhoa1 decoding for HF.




(c) Gains as a function of the angle, for the HF IDHOA decoder.

Figure 4.5: First order decoding to 2.0 layout, obtained with IDHOA software.

2. *Panning test.* Circular panning, one round and two rounds in ten seconds. Reference is a standard amplitude panning with size (maximum cross-talk with adjacent channels is approximately -12 dB). The speakers layout of the reference, in the absence of a rotating loudspeaker, has been designed to be a custom 8.0 setup (5.0 with three more channels at $\pm 90^{\circ}$ and $180^{\circ}$).

3. *Global perception test.* Custom object-based mix of a pop song rendered through 5.0 Ambisonics and 8.0 reference layout with amplitude panning.

For the first two tests the types of sources used are broadband noise (pink noise), voice (male English voice speaking, recorded in anechoic room) and music (fragment of flamenco with voice and instruments).

Each subject had to evaluate 7 different dual-band decoders, with cut-off frequency set to 400 Hz, (5 Ambisonics, 1 reference, 1 anchor) with respect to the reference. The anchor has been chosen to be a basic single-band "naïve" or "projection" decoding.

For test 1 (localization test), each subject evaluated the 7 decoders in 9 trials (3 positions times 3 signals). For test 2 (panning test), each subject evaluated the 7 decoders in 4 trials (2 pannings times 2 signals). Test 3 (assessing the global spatial perception), had only one trial per subject.

The tests where in one trial the reference is evaluated less than 90 over 100 are discarded from the analysis, leaving 11 subjects in the worst case.

The three tests were done in succession and lasted between 30 minutes and 2 hours, depending on the listener.

For each one of the three tests we do the following comparisons:

1. BHL1 vs. idhoa1

2. FAA2 vs. idhoa2

3. idhoa2 vs. idhoa1

badly. For this reason we concentrated on positions where a loudspeaker is present in the Ambisonics setup.

4. idhoa3 vs. idhoa2

After checking the normality of the data with the Kolmogorov-Smirnov test, each pairwise comparison is done using the two-tailed paired t-test method on the averages of the trials of each subject. The statistical influence of multiple comparisons is considered and corrected with the Holm-Bonferroni method [Abdi, 2010].

### 4.2.2 Tests Results

Figures 4.6 and 4.7 show the results of the three different tests. Figures 4.6(a) and 4.6(b) show the results of the localization test, grouped for source direction and source type, respectively.

In general the BLH1 decoder exhibits good properties at localizing the sources in correct place [e.g. see the 110° set of trials in Figure 4.6(a)] but the lower loudness, due to the pressure normalization at high frequencies, was negatively evaluated by all listeners. Globally, the BLH1 decoder was not better evaluated than the anchor, probably due to the loudness issue.

The FAA2 decoder suffers especially when the source is at 110°. This suggests that the request for zero angular error in every direction at the expense of spatial sharpness and crosstalk, as FAA2 does, is detrimental to localization performances.

Some decoders, particularly BHL1 and idhoa1, perform especially bad at 30°. Particularly problematic for Ambisonics is the frontal region at 0°, where all the three speakers are active at the same time.

In the global analysis, averaging all the measurements, it is possible to highlight a trend for the decoders where idhoa1 and FAA2 are almost equivalent, and idhoa2 and 3 are better evaluated than the former.

All the listeners reported that the differences between the decoders where evident when listening to the broadband noise, while much more subtle when using "natural" signals, especially music. Figures 4.6(b) and 4.7(a) show that the MUSHRA scores are higher for voice and music than for noise, both in localization and panning.

Since the Kolmogorov-Smirnov test showed no significant deviations from normality, data are analyzed performing a pairwise comparison between four

| | Test 1 (localization) | | | |
|---|---|---|---|---|
| Comparison | Orig. $p$-value | Corr. $p$-value | S | Diff. |
| idhoa1 vs. BHL1 | 0.004 | 0.008 | ** | 14 |
| idhoa2 vs FAA2 | 0.000014 | 0.00006 | *** | 14 |
| idhoa2 vs idhoa1 | 0.002 | 0.007 | ** | 8 |
| idhoa3 vs idhoa2 | 0.09 | 0.09 | – | 2 |

| | Test 2 (panning) | | | |
|---|---|---|---|---|
| Comparison | Orig. $p$-value | Corr. $p$-value | S | Diff. |
| idhoa1 vs. BHL1 | 0.04 | – | – | 16 |
| idhoa2 vs FAA2 | 0.1 | – | – | 9 |
| idhoa2 vs idhoa1 | 0.04 | 0.08 | – | 6 |
| idhoa3 vs idhoa2 | 0.11 | – | – | 3 |

| | Test 3 (global perception) | | | |
|---|---|---|---|---|
| Comparison | Orig. $p$-value | Corr. $p$-value | S | Diff. |
| idhoa1 vs. BHL1 | 0.0008 | 0.003 | ** | 23 |
| idhoa2 vs FAA2 | 0.4 | – | – | −2 |
| idhoa2 vs idhoa1 | 0.3 | 0.96 | – | 4 |
| idhoa3 vs idhoa2 | 0.98 | – | – | 0 |

Table 4.3: Significance analysis of the four comparisons in the three tests. The original $p$-value lists the result of the two-tail paired t-test. The corrected value corresponds to the result of the Holm-Bonferroni correction. The column "S" indicates the significance. The "Diff." column indicates the average difference in MUSHRA points.

combinations of decoders using the two-tailed paired t-test method, as explained in Section 4.2.1, and results are summarized in Table 4.3.

In the localization test, the pairwise comparison reveals that there is significant difference between idhoa1 and BHL1, idhoa2 and FAA2, where the IDHOA decoders are significantly better rated than the alternatives. When comparing idhoa2 and idhoa1, the former gets significantly better evaluation than the latter. While checking for idhoa3 against idhoa2, no significant difference is found. Not surprisingly, this trend follows closely the values for $I_{\mathrm{R}}$ reported in Table 4.1.

In the panning test none of the four comparisons gives significant difference among the decoders. Nevertheless, the tendency is completely analogous to the localization test, hinting that similar significant results could perhaps be obtained with increased statistics.

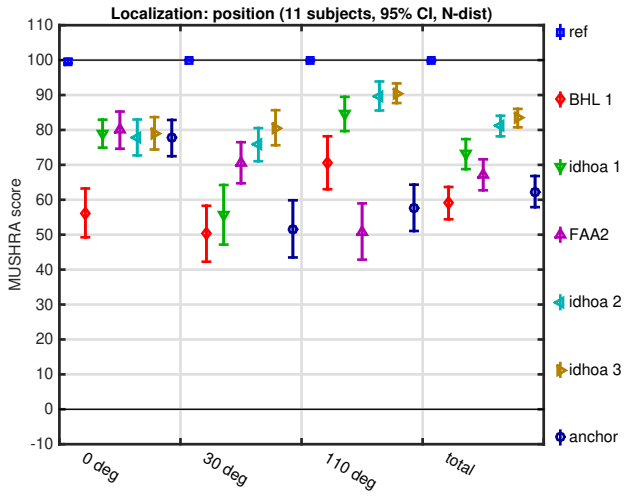
For the global evaluation test only the comparison idhoa1 versus BHL1 results significant, and the former is significantly better rated than the latter. Again, this could be due to the level difference.

## 4.3 Summary

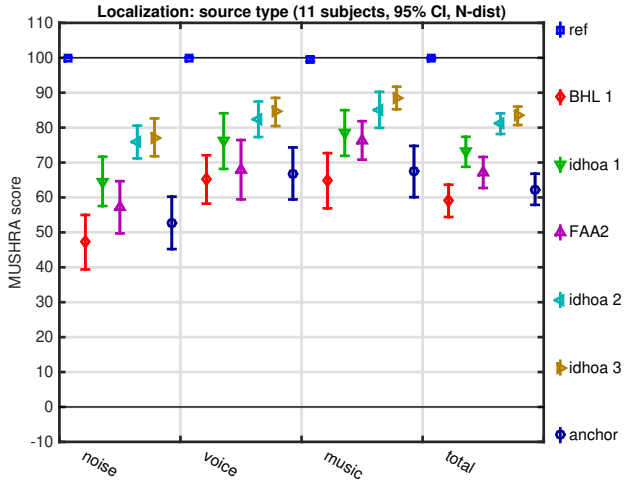
IDHOA can produce a wide variety of decoders both in 2D and 3D, allowing for a fine control over the loudness and localization properties by tuning a small set of parameters.

The energy and intensity plots show that the decodings generated with IDHOA have –to some extent– better directionality properties than the state-of-the-art decoders at the expense of some error in localization.

Subjective testing has shown that localization properties of the decodings generated by IDHOA are better evaluated than the state-of-the-art decoders, with a similar trend for the panning properties (although results are not significant in this case). On the other hand, no significant differences have been found in the global evaluation test (except for the state-of-the-art first order decoder, which can probably be attributed to a level mismatch). This might indicate that the chosen fragment is not representative enough to show differences between the decoders.

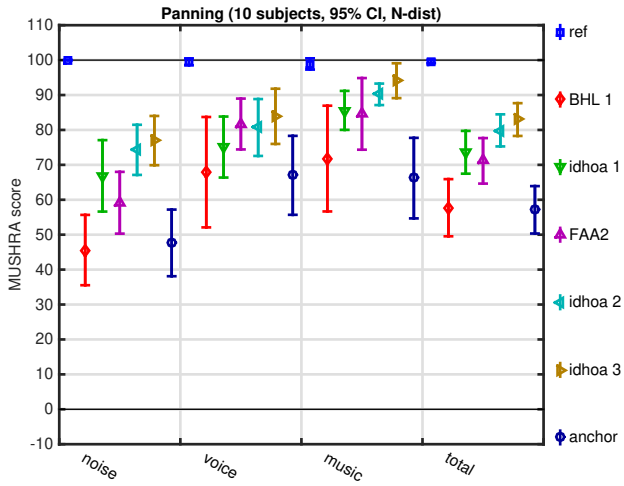



(a) Test 1 (localization). Trials grouped for source direction.

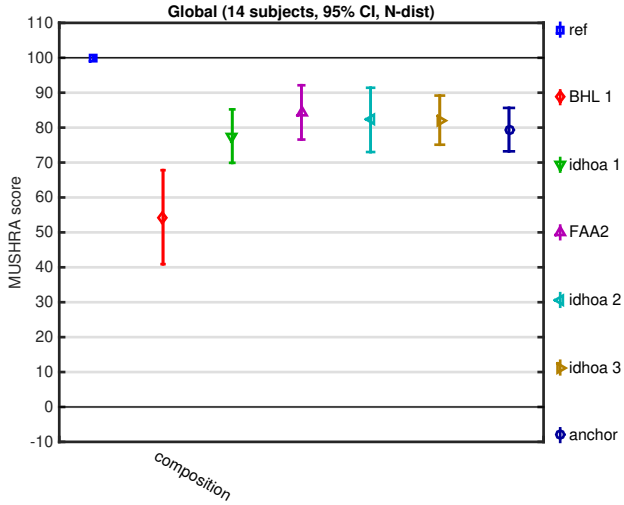



(b) Test 1 (localization). Trials grouped for source type.

Figure 4.6: Listening tests results. Figure (a) shows the listening test scores for source position and size evaluation grouped for source position, while in (b) the scores are grouped for source type. Error bars correspond to two times the standard deviation of the mean.




(a) Test 2 (panning). Trials grouped for source type.




(b) Test 3 (global perception).

Figure 4.7: Listening tests results. Figure (a) shows the results for panning quality grouped for source type. Figure (b) reports the scores obtained by the different decoders evalueted with a "pop song" spatial composition. Error bars correspond to two times the standard deviation of the mean.

The code used to generate the Ambisonics decodings and the decodings themselves are publicly available in [Scaini, 2015].

# Part II

# Wavelets

# Chapter 5

# INTRODUCTION TO WAVELET THEORY

In this Chapter we want to give sufficient background information to be able to understand the idea behind our wavelet optimization, even if our approach aims at capturing the main concepts of wavelets, e.g. locality, more than a formal wavelet construction.

The contributions to Wavelet Theory come from very different areas of Science and Engineering. Because the wavelets come from very different areas of expertise, there are many ways to motivate their construction and understand their properties. This fragmented and diverse development also lead to many wavelet transforms and wavelet-generation schemes. Moreover, one of the main concepts behind the wavelet transforms, especially for compression, is to adapt the basis of the analysis functions to the signal to be analyzed. For this reason there are almost as many wavelet families as applications or problems, increasing the wild diversity of wavelet approaches.

## 5.1 Introduction to Wavelet Transforms

In this section we will gradually introduce the concept of Wavelet transform by similarity and difference with the Fourier transforms.

**Fourier Transform** The Fourier transform (FT) of a one dimensional square integrable signal $s(t)$ is given by

$$S(f) = \int_{-\infty}^{+\infty} s(t) e^{-2i\pi ft} dt \tag{5.1}$$

the inverse transform is given by

$$s(t) = \int_{-\infty}^{+\infty} S(f) e^{2i\pi ft} df. \tag{5.2}$$

Equation (5.1) gives a representation of the frequency content of the signal $s(t)$ but gives no information about its localization in time, vice versa for Eq. (5.2). The bases of the Fourier transform are the sine and cosine. The FT, then, maps time (or space) to frequency (and vice versa) but, because of the infinite support of the FT basis functions, it is impossible to have information on time and frequency at the same time. Note that the same happens with the Spherical Harmonics (and so with Ambisonics), just the two domains connected by the SH are space, $(\theta, \phi)$, and 'frequency' in spherical harmonics terms, $(l, m)$.

**Windowed Fourier Transforms** One way to have information on both domains at the same time while preserving the linearity of the operator is to introduce a window, giving birth to the windowed Fourier transform (WFT), also known as short-time Fourier transform (STFT). Being $w(t)$ a window function (real, for simplicity) with a finite integral and compact support (i.e. non-zero over a finite interval[1]), the WFT of the signal $s(t)$ is defined as:

$$S_W(\tau, f) = \int_{-\infty}^{+\infty} s(t) w(t-\tau) e^{-2i\pi ft} dt$$

The application of a window has several consequences. The transform is function of two variables, the frequency $f$ and the position at which the

[1] This condition is often relaxed by asking some fast decay, for example exponential.

window is applied $\tau$. The filter function $w(t)$ is a window in time but also a window in the frequency spectrum $f$ around $\tau$. The shape of the filter in frequency domain is $W(f)$, which is the FT of $w(t)$. One thing that is often forgotten (and it is good to keep in mind also when we will talk about wavelets) is that the shape of $W(f)$ in general is very different from $w(t)$, and only for a limited set of functions it is possible to get $w(t) \propto W(f)$. The choice of the window in the time domain affects the shape of the window in frequency domain: typically there will be a main lobe and some "spill" at low and high frequencies. If we define the spread in frequency, *bandwidth*, of the window $w(t)$ as:

$$\Delta f^2 = \frac{\int f^2 |W(f)|^2 df}{\int |W(f)|^2 df},$$

while the spread in time can be defined as:

$$\Delta t^2 = \frac{\int t^2 |w(t)|^2 dt}{\int |w(t)|^2 dt}$$

(by Parseval's theorem both denominators are equal, and are the energy of $w(t)$) then the Heisenberg inequality bounds their product[2]

$$\Delta f \, \Delta t \geq \frac{1}{4\pi}.$$

The Heisenberg inequality has two consequences. First, it is not possible to have infinite precision in both time and frequency. This means that it is not possible to separate two impulses that are closer than $\Delta t$ or separate two tones that are closer than $\Delta f$. The second effect is that, once the window is chosen, the resolution limit is the same over all times and frequencies. For example, let's imagine we choose a window in time to have good resolution at mid frequencies, then we will get poor resolution in frequency for the low frequencies, but for high frequencies we will get very good resolution in frequency and

[2]The lower bound $\Delta f \, \Delta t = \frac{1}{4\pi}$ is reached by the Gabor transform that uses a Gaussian as $w(t)$ window function. Note that the FT of a Gaussian is a Gaussian, so the $W(f)$ is a Gaussian too. The Gabor transform has a set of nice properties that come from the choice of the Gaussian as $w(t)$.

a very bad one in time. (In typical audio applications more than one STFT is run in parallel with different window sizes).

**Wavelet Transform** As already said, the paths that lead to the Wavelet Transform come from very different directions, but the ideas and motivations behind it are the same and can be reduced to two:

1. constant resolution along frequency: $\frac{\Delta f}{f} = c$, with $c$ a constant (in signal processing is known as *constant-Q* analysis);

2. to model a signal use a basis that is similar to the signal resulting in less coefficients and better compression.

The first concept is visually rendered in Figure 5.1: instead of changing the frequency of the basis function inside a window of fixed length, the idea is to compress or stretch (scale) a time-limited oscillating function effectively changing its support and frequency at the same time.

If we call $\psi$ this "time-limited oscillating function" then we can write this idea as:

$$CWT_s(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{+\infty} s(t)\psi^* \left(\frac{t-b}{a}\right) dt. \tag{5.3}$$

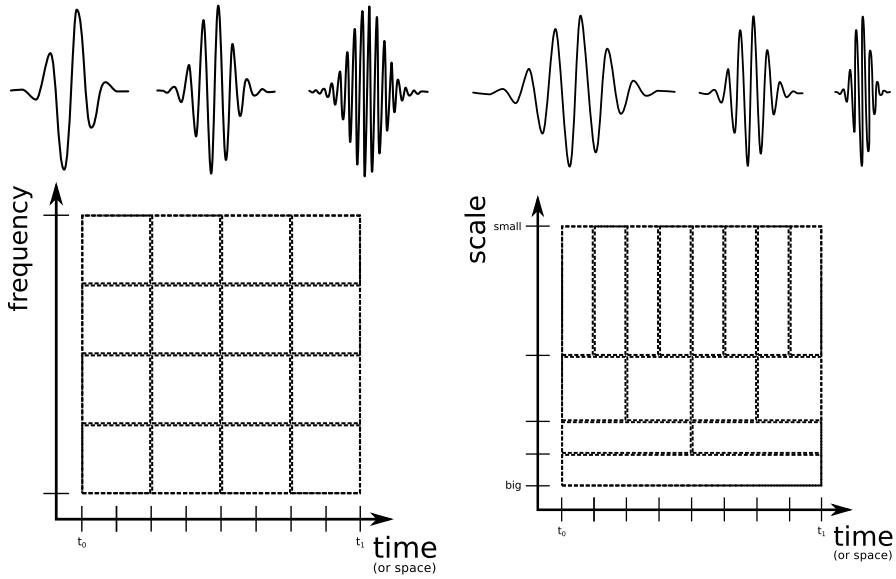
$CWT_s(a,b)$ is the Continuous Wavelet Transform (CWT) of the signal $s(t)$ and is function of two variables: $a$, called *dilation* (scale) and $b$, the *translation*. Dilating a wavelet means stretching it (if $|a| < 1$) or compressing it (if $|a| > 1$). We can restrict to $a > 0$ without loss of generality.

Typically Eq. (5.3) is written in a more compact form as:

$$CWT_s(a,b) = \int_{-\infty}^{+\infty} s(t)\psi^*_{a,b}(t)dt$$

with

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right), \text{with } a, b \in \mathbb{R}$$




(a) STFT - The oscillations increase in frequency inside a window of fixed length.

(b) WT - The oscillating function is compressed/stretched (scaled or dilated).

Figure 5.1: Simplified illustration of the STFT fixed window paradigm, versus the idea of dilation and scale in WT. The idea for the illustrations in the upper row is taken from [Barford et al., 1992].

being the *mother wavelet*, which has to satisfy a couple of properties. The first is called the *admissibility condition*

$$\int \psi(t)dt = 0,$$

and the second property asks for $\psi$ to be square integrable (i.e. have finite energy)

$$\int \psi^2(t)dt < \infty.$$

The admissibility condition is often reported for the Fourier tranform of $\psi(t)$, $\Psi(\omega)$, and translates into

$$\int \frac{|\Psi(\omega)|}{|\omega|} d\omega < +\infty.$$

with $\omega = 2\pi f$. The admissibility condition implies that $\Psi(\omega)$ vanishes at the zero frequency, (otherwise the integral would blow up at $\omega = 0$)

$$|\Psi(\omega)^2|_{\omega=0} = 0.$$

This means that the wavelets must have a band-pass like spectrum.

The inverse formula, the Inverse Continuous Wavelet Transform (ICWT), is given by:

$$s(t) = \frac{1}{c_\psi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (CWT_s)_{a,b} \psi_{a,b}(t) \frac{dadb}{a^2} \tag{5.4}$$

where $c_\psi = 2\pi \int_{-\infty}^{+\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega$, and $\Psi(\omega)$ is the Fourier transform of $\psi(t)$. Equation (5.4) can be interpreted in two ways:

- As a way of reconstructing $s(t)$ once its wavelet transform $(CWT_s)_{a,b}$ is known; this formula is known as the *reconstruction formula or scheme* (or *resolution of the identity*)

- As a way to write $s$ as a superposition of wavelets $\psi_{a,b}$. the coefficients in this superposition are exactly given by the wavelet transform of $s$.

Note that the $\psi_{a,b}$ are defined over every point in the $(a, b)$ space, and so they are highly redundant. Is it possible to discretize the $(a, b)$ space so that the $\psi_{a,b}$ form a true orthonormal basis?

**Discretized Wavelet Transform** Let's start from discretizing the dilation parameter $a$ as a power of a fixed dilation step $a_0 > 1$: $a = a_0^m$, with $m \in \mathbb{Z}$. The parameter $m$ will control the dilation, while the translation will be $b = nb_0a_0^m$ with $n \in \mathbb{Z}$, so that is adapted to the width of the wavelet. This gives

$$\psi_{m,n}(t) = \psi_{(a_0^m, nb_0a_0^m)}(t) = a_0^{-m/2}\psi(a_0^{-m}t - nb_0) \tag{5.5}$$

and the discretized wavelet coefficients are

$$d_{m,n} = \int s(t)\psi^*_{m,n}(t)dt.$$

The choice of the $a_0$ and $b_0$ parameters defines the type of the wavelet family. A common choice that goes under the name of *dyadic sampling* is $a_0 = 2$ and $b_0 = 1$, giving dyadic sampling along frequency and time respectively. The question translates now in if the $\psi_{m,n}$ and the inverse wavelet transform

$$\tilde{s}(t) = c\sum_{m,n} d_{m,n}\psi_{m,n}(t) \tag{5.6}$$

form a sort of discrete approximation[3] of Eq.(5.4), so that $\tilde{s}(t) \approx s(t)$. The short answer is *yes*: it is possible to design some $\psi_{m,n}$ so that the Eq. (5.6) is actually an equation that defines the Discrete Wavelet Transform (DWT). The important thing to notice is that the wavelet transform can be redundant (when $\psi_{m,n}$ is a frame[4]) or not (when $\psi_{m,n}$ is a basis), and this redundacy

[3] In math terms if a family of wavelets $\psi_{m,n}$ constitutes a *frame*.

[4] A set of non-zero vectors $\{\phi_i\}_{i\in J}$ constitutes a frame in the Hilbert space $\mathcal{H}$, if exist an $A > 0$ and a $B < \infty$ such that, for all $f \in \mathcal{H}$: $A\|f\|^2 \leq \sum_{i\in J}|\langle\phi_i|f\rangle|^2 \leq B\|f\|^2$. When $A = B$ the frame is called a tight frame.

can be (somewhat) tuned and can be actually an interesting feature for signal analysis. It is also interesting to exploit this redundancy in (numerical) reconstruction, because for a given reconstruction precision, the redundacy allows to calculate the wavelet coefficients with less precision than the one needed with zero redundacy (othonormal bases), at the cost of having more coefficients, [Daubechies, 1990].

Before reaching a more modern way to build wavelets (via the *multiresolution analysis*), we have to introduce a couple of notions, at least intuitively. The first is the concept of *scaling function* (or *smoothing function*), $\phi(t)$, introduced by Mallat [Mallat, 1989]. If we say that $m = 0$ is the lowest value for the dilation parameter or, in other words, the lowest *level* at which we are decomposing the signal $s(t)$, we need something that takes what remains of $s(t)$ at the point we stopped the decomposition. Since the wavelet functions are bandpass like filters, we need a low-pass kind of function. The filter that fulfills this role is the scaling function $\phi(t)$ and has the property that $\int \phi(t)dt = 1$. Very much like for wavelets, (5.5), the scaling functions families are also dilated and translated copies of a an original scaling function

$$\phi_{m,n}(t) = a_0^{-m/2}\phi(a_0^{-m}t - nb_0), \text{ with } n \in \mathbb{Z}.$$

The Eq. (5.6), in this context, could become something like this

$$\tilde{s}(t) = c_{0,0}\phi_{0,0}(t) + \sum_{m=0}^{J-1}\sum_{n\in\mathbb{Z}} d_{m,n}\psi_{m,n}(t)$$

with $J$ the maximum level of decomposition.

The second concept is the distinction between the "First Generation wavelets" and the "Second Generation wavelets". The main differences are two, one is about the relation between scaled/translated wavelets and the other concerns the framework to actually build the wavelet filters.

Regarding the relation between scaled/translated wavelets, Schröder and Sweldens explain perfectly in [Schröder and Sweldens, 1995] the shift in paradigm from the wavelet scheme described in this Section, called First Generation wavelets, to the Second Generation:

> "In the classic wavelet setting, i.e., on the real line, wavelets are defined as the dyadic translates and dilates of one particular, fixed function. They are typically built with the aid of a scaling function. Scaling functions and wavelets both satisfy refinement relations (or two scale relations). This means that a scaling function or wavelet at a certain level of resolution ($j$) can be written as a linear combination of scaling basis functions of the same shape but scaled at one level finer (level $j+1$)[5] [...] The basic philosophy behind second generation wavelets is to build wavelets with all desirable properties (localization, fast transform) adapted to much more general settings than the real line. [...] Adaptive constructions rely on the realization that translation and dilation are not fundamental to obtain the wavelets with the desired properties. The notion that a basis function can be written as a finite linear combination of basis functions at a finer, more subdivided level, is maintained and forms the key behind the fast transform. The main difference with the classical wavelets is that the filter coefficients of second generation wavelets are not the same throughout, but can change locally to reflect the changing (non translation invariant) nature of the surface and its measure."

Concerning the actual method to operatively build the wavelets, they say:

> The tool that we use to build wavelets transforms is called the *lifting scheme*. The main feature of the lifting scheme is that all constructions are derived in the *spatial domain*. This is in contrast to the traditional approach wich relies heavily on the frequency domain[6]. Staying in the spatial domain leads to two major advantages. First, it does not require the machinery of Fourier analysis as a prerequisite. This leads to a more intuitively appealing treatment better suited to those interested in applications, rather

[5] To make the dissertation more agile, we give a universal definition of the refinement relations (that works for first and second generation wavelets) in Section 5.2, instead of following the historical development of the theory.

[6] See for example [Daubechies, 1992].

> than mathematical foundations. Secondly, lifting leads to algorithms that can easily be generalized to complex geometric situations which typically occur in computer graphics. This will lead to so called *Second Generation Wavelet*. [...] Even though the wavelets which result from using the lifting scheme in the more general settings will not be translates and dilates of one function anymore they still have all the powerful properties of first generation wavelets: fast transforms, localization and good approximation.

The operative construction of second generation wavelets via the lifting scheme is beautifully described in the same manuscript [Sweldens and Schröder, 1995]. For an exhaustive mathematical definition of the lifting scheme, the interested reader should refer to [Sweldens, 1998]. We will give a concise definition in Section 5.4.

With this brief, and possibly agile, introduction we can move to a proper and more modern definition of wavelet transforms.

## 5.2 Multiresolution Formulation with Matrix Notation

The starting point to define wavelets is a mathematical framework called *multiresolution analysis*. To define the multiresolution analysis, we have to define first a nested set of closed vector subspaces $V^0 \subset \cdots \subset V^j \subset \cdots \subset V^n$. The higher the space index, the finer is the space. For each $j$, the basis functions of $Vj$ are called *scaling functions*, and are denoted like this: $\phi_k^j$ with $k \in \mathbb{K}(j)$, where $\mathbb{K}$ is and index set with $\mathbb{K}(j) \subset \mathbb{K}(j+1)$. Since the vector spaces are nested, it is possible to write each $\phi_k^j$ as a function of the next level $\phi_{j+1}$, and obtain these *refinement relations*:

$$\phi_k^j = \sum_l p_{l,k}^{j+1} \phi_l^{j+1}$$

where $n > j \geq 0$, $k \in \mathbb{K}(j)$ and $l \in \mathbb{K}(j+1)$. Note: in this refinement relation there is no explicit mention to how dilation and translation are implemented, e.g. (5.5). This definition is valid also in the second generation wavelets, where the dilation and translation relations are not maintained between different levels. Additionally, the vector spaces used to build the multiresolution are very generic. The construction of wavelets in the multiresolution framework follows the same procedure for 1D, 2D or n-dimensional spaces.

Adopting a more compact and convenient matrix notation, putting together the different scaling functions $\phi_k^j$ for the level $j$ as one row vector:

$$\Phi^j = \begin{pmatrix} \phi_1^j & \cdots & \phi_{m^j}^j \end{pmatrix}$$

where $m^j$ is the dimension of $V^j$ (here we assume that $V^j$ has a finite basis).

The *wavelet spaces*, $W^j$, are defined to be the orthogonal complement of $V^j$ in $V^{j+1}$, so that $V^j \oplus W^j = V^{j+1}$. Meaning that $W^j$ includes all the functions in $V^{j+1}$ that are orthogonal to all those in $V^j$ under some inner product (typically $\mathcal{L}^2$). The functions that form a basis of $W^j$ are called *wavelets*, and are denoted with $\psi_p^j$. The corresponding refinement equations for the wavelets are as follows:

$$\psi_k^j = \sum_l q_{l,k}^{j+1} \phi_l^{j+1}$$

Similarly to the scaling functions, we can group them in a row vector:

$$\Psi^j = \begin{pmatrix} \psi_1^j & \cdots & \psi_{n^j}^j \end{pmatrix}$$

where $n^j$ is the dimension of $W^j$, with $m^{j+1} = m^j + n^j$.

With this matrix notation it is possible to rewrite the *refinement relations*:

$$\Phi^j = \Phi^{j+1} \mathbf{P}^{j+1}, \tag{5.7}$$

and, in similar fashion, a matrix $\mathbf{Q}$ must exist to satisfy:

$$\Psi^j = \Phi^{j+1} \mathbf{Q}^{j+1}. \tag{5.8}$$

The biorthogonality conditions then become

$$\langle \Phi^j | \Psi^j \rangle = \mathbf{0} \tag{5.9}$$

where

$$\langle \Phi^j | \Psi^j \rangle_{kl} = \langle \phi^j_k | \psi^j_l \rangle,$$

and $\langle \phi | \psi \rangle$ denotes the inner product. Substituting (5.8) into this last equation (5.9), it gives

$$\langle \Phi^j | \Phi^{j+1} \rangle \mathbf{Q}^{j+1} = \mathbf{0}$$

this is an homogeneous system of linear equations and there is no unique solution to it. The matrix $\mathbf{Q}$ is a basis of the space of all possible solutions. So there is no unique $\mathbf{Q}$, meaning that there are many different wavelet bases for a given wavelet space $W^j$. To determine uniquely the $\mathbf{Q}$ matrices we have to impose further constraints to the orthogonality alone. The discussion to the different constraints options and resulting wavelets available in literature is out of the scope of this introduction to wavelets, more information can be found in [Stollnitz et al., 1995].

Each multiresolution analysis is accompanied by a dual multiresolution analysis consisting of nested spaces $\tilde{V}^j$ with bases given by dual scaling functions $\widetilde{\Phi}^j$, which are biorthogonal to the scaling functions:

$$\langle \widetilde{\Phi}^j | \Phi^j \rangle = \mathbf{1}.$$

The dual scaling functions satisfy similar *refinement relations* as of Eq. (5.7):

$$\widetilde{\Phi}^j = \widetilde{\Phi}^{j+1} [\mathbf{A}^{j+1}]^T. \tag{5.10}$$

Similarly, for any given wavelet basis there is a dual basis $\widetilde{\Psi}^j$ and the two are biorthogonal with respect to each other: $\langle \widetilde{\Psi}^j | \Psi^j \rangle = \mathbf{1}$. This also implies $\langle \widetilde{\Psi}^j | \Phi^j \rangle = \langle \widetilde{\Phi}^j | \Psi^j \rangle = \mathbf{0}$. And similarly to Eq. (5.8), a matrix $B$ will exists, so that:

$$\widetilde{\Psi}^j = \widetilde{\Phi}^{j+1} [\mathbf{B}^{j+1}]^T. \tag{5.11}$$

Combining the fact that $\phi_k^j \in V_j \oplus W_j$ with the biorthonormality relations leads to the inverse refinement equations for the original scaling function:

$$\Phi^{j+1} = \Phi^j \mathbf{A}^{j+1} + \Psi^j \mathbf{B}^{j+1}. \tag{5.12}$$

The scaling function coefficients $\mathbf{c}^j$ and wavelet coefficients $\mathbf{d}^j$ of any function $f$ can be obtained by inner product with the dual scaling function and dual wavelets respectively:

$$\mathbf{c}^j = \langle \widetilde{\Phi}^j | f \rangle,$$
$$\mathbf{d}^j = \langle \widetilde{\Psi}^j | f \rangle.$$

These operators $\mathbf{A}^j, \mathbf{B}^j$ and $\mathbf{P}^j, \mathbf{Q}^j$ are the *decomposition* and *reconstruction filters* (respectively). The biorthogonality properties imply that the operators $\mathbf{A}^j$, $\mathbf{B}^j$, $\mathbf{P}^j$ and $\mathbf{Q}^j$ need to verify the following properties:

$$\langle \widetilde{\Psi}^j | \Phi^j \rangle = \mathbf{0} \implies \mathbf{B}^j \mathbf{P}^j = \mathbf{0}, \tag{5.14a}$$
$$\langle \widetilde{\Phi}^j | \Psi^j \rangle = \mathbf{0} \implies \mathbf{A}^j \mathbf{Q}^j = \mathbf{0}, \tag{5.14b}$$
$$\langle \widetilde{\Phi}^j | \Phi^j \rangle = \mathbf{1} \implies \mathbf{A}^j \mathbf{P}^j = \mathbf{1}, \tag{5.14c}$$
$$\langle \widetilde{\Psi}^j | \Psi^j \rangle = \mathbf{1} \implies \mathbf{B}^j \mathbf{Q}^j = \mathbf{1} \tag{5.14d}$$

and can be rewritten in matrix notation like:

$$\begin{bmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{bmatrix} \begin{bmatrix} \mathbf{P}^j & \mathbf{Q}^j \end{bmatrix} = \begin{bmatrix} \mathbf{AP} & \mathbf{AQ} \\ \mathbf{BP} & \mathbf{BQ} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \text{ and } \begin{bmatrix} \mathbf{P}^j & \mathbf{Q}^j \end{bmatrix} \begin{bmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{bmatrix} = \mathbf{1} \tag{5.15}$$

Additionally, applying the biorthogonality properties to (5.12) it means that:

$$\mathbf{P}^j \mathbf{A}^j + \mathbf{Q}^j \mathbf{B}^j = \mathbf{1}. \tag{5.16}$$

Equations (5.14) and (5.16) can be combined by stating that the decomposition and reconstruction filters are globally inverse to the other:

$$\begin{pmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{pmatrix} = \begin{pmatrix} \mathbf{P}^j & \mathbf{Q}^j \end{pmatrix}^{-1}.$$

## 5.3 Subdivision Mesh

Now we are going to introduce a concept that is crucial for the practical construction of wavelets throughout the rest of the thesis. A *subdivision mesh* is a method of representing a smooth surface (in this case, the sphere) as the limit of a series of increasingly finer polygonal meshes. The mesh is built recursively starting from a primitive polygonal mesh (e.g. an octahedron), and subdividing (i.e. adding new vertices) this original mesh according to some rule, called subdivision scheme. Examples of subdivision schemes are Loop [Loop, 1987], Catmull-Clark [Catmull and Clark, 1978], Doo-Sabin [Doo, 1978]. At each iteration a finer (more dense) mesh is obtained. In this work we will call each iteration a *level*, and the primitive mesh is referred to as *level 0*. The wavelet framework is built out of the subdivision mesh. A given wavelet level will be associated to a certain mesh level. We will only consider subdivision meshes up to some given order $n$, corresponding to the finest mesh.

To close the circle, when the function space is a finite vector space defined over the finest space $V^n$, and the dimensionality of the function space coincides with the dimensionality of the finest mesh, the scaling functions at the finest level $n$ can be taken to delta functions: $\phi_k^n(p) = \delta(p-k)$. In this case the wavelets and dual wavelets can be computed from the decomposition and reconstruction filters at each level as follows:

$$\begin{aligned}
\phi_k^j(p) &= \left(\mathbf{P}^n \cdots \mathbf{P}^{j+2}\,\mathbf{P}^{j+1}\right)_{pk} \\
\psi_k^j(p) &= \left(\mathbf{P}^n \cdots \mathbf{P}^{j+2}\,\mathbf{Q}^{j+1}\right)_{pk} \\
\tilde{\phi}_k^j(p) &= \left(\mathbf{A}^{j+1}\,\mathbf{A}^{j+2} \cdots \mathbf{A}^n\right)_{kp} \\
\tilde{\psi}_k^j(p) &= \left(\mathbf{B}^{j+1}\,\mathbf{A}^{j+2} \cdots \mathbf{A}^n\right)_{kp}
\end{aligned} \tag{5.17}$$

In fact, by using the procedural approach that we will outline in Section 6.2, it is perfectly possible to ignore the scaling functions, the wavelets and their duals and work only with the scaling and wavelet coefficients and the decomposition and reconstruction filters.

## 5.4 Second Generation Wavelets via the Lifting Scheme

From the construction of wavelets in the scale and dilate paradigm we introduced in Section 5.1 to the method we implemented for our specific application, that will be discussed in Chapter 7, there is quite a leap forward in methods and meanings. To cover this distance we introduce a procedural method to build wavelets, the Lifting Scheme.

### 5.4.1 Lifting Scheme

Given an initial set of bihortogonal filter operators $\{\bar{\mathbf{A}}^j, \bar{\mathbf{P}}^j, \bar{\mathbf{B}}^j, \bar{\mathbf{Q}}^j\}$, then a new set of bihortogonal filters $\{\mathbf{A}, \mathbf{P}, \mathbf{B}, \mathbf{Q}\}$ can be found as

$$\begin{aligned} \mathbf{P}^j &= \bar{\mathbf{P}}^j \\ \mathbf{A}^j &= \bar{\mathbf{A}}^j + \mathbf{S}^j \bar{\mathbf{B}}^j \\ \mathbf{Q}^j &= \bar{\mathbf{Q}}^j - \bar{\mathbf{P}}^j \mathbf{S}^j \\ \mathbf{B}^j &= \bar{\mathbf{B}}^j \end{aligned} \tag{5.18}$$

Indicating explicitly the dimensions of the operators, as in Section 6.1.3,

$$\begin{aligned} \mathbf{P}^j_{M\times K} &= \bar{\mathbf{P}}^j_{M\times K} \\ \mathbf{A}^j_{K\times M} &= \bar{\mathbf{A}}^j_{K\times M} + \mathbf{S}^j_{K\times(M-K)} \bar{\mathbf{B}}^j_{(M-K)\times M} \\ \mathbf{Q}^j_{M\times(M-K)} &= \bar{\mathbf{Q}}^j_{M\times(M-K)} - \bar{\mathbf{P}}^j_{M\times K} \mathbf{S}_{K\times(M-K)} \\ \mathbf{B}^j_{(M-K)\times M} &= \bar{\mathbf{B}}^j_{(M-K)\times M} \end{aligned}$$

It is quite straightforward to prove writing the lifting scheme in matrix notation (we assume for simplicity that all matrices are real valued)

$$\begin{bmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{S}^j \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{A}}^j \\ \bar{\mathbf{B}}^j \end{bmatrix}$$

and

$$\begin{bmatrix} \mathbf{P}^j \\ \mathbf{Q}^j \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\mathbf{S}^j & \mathbf{1} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{P}}^j \\ \bar{\mathbf{Q}}^j \end{bmatrix}$$

and if we recall the bihorthogonality relations (5.15), we get

$$\begin{aligned} \begin{bmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{bmatrix} \begin{bmatrix} \mathbf{P}^j & \mathbf{Q}^j \end{bmatrix} &= \begin{bmatrix} \mathbf{1} & \mathbf{S}^j \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{A}}^j \\ \bar{\mathbf{B}}^j \end{bmatrix} \begin{bmatrix} \bar{\mathbf{P}}^j & \bar{\mathbf{Q}}^j \end{bmatrix} \begin{bmatrix} \mathbf{1} & -\mathbf{S}^{j\top} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{1} & \mathbf{S}^j \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{1} & -\mathbf{S}^{j\top} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \qquad \blacksquare \end{aligned}$$

From the relations (5.7) (5.8) (5.10) (5.11) and the definition of lifting scheme (5.18), it is possible to see how the lifting scheme impacts the scaling functions, the wavelets, and their duals:

$$\begin{aligned} \Phi^j &= \bar{\Phi}^j \\ \widetilde{\Phi}^j &= \bar{\mathbf{A}}^j \widetilde{\Phi}^{j+1} + \mathbf{S}^j \bar{\mathbf{B}}^j \widetilde{\Phi}^{j+1} = \bar{\mathbf{A}}^j \widetilde{\Phi}^{j+1} + \mathbf{S}^j \widetilde{\Psi}^j \\ \Psi^j &= \bar{\mathbf{Q}}^j \Phi^{j+1} - \bar{\mathbf{P}}^j \mathbf{S}^j \Psi^{j+1} = \bar{\Psi}^j - \mathbf{S}^j \bar{\Phi}^j \\ \widetilde{\Psi}^j &= \bar{\mathbf{B}}^j \widetilde{\Phi}^j \end{aligned}$$

A note of caution: the notation we are using while being convenient, because it is extremely compact, it partially hides the inner workings of $\mathbf{S}$. For an explicit element-by-element definition of the lifting scheme (*index notation*), we refer to [Sweldens, 1998].

The great benefit of the lifting scheme is that, starting from some simple or even trivial filters $\{\bar{\mathbf{A}}^j, \bar{\mathbf{P}}^j, \bar{\mathbf{B}}^j, \bar{\mathbf{Q}}^j\}$ it is possible to build more complex ones just by tuning the operator $\mathbf{S}^j$. By controlling the operator $\mathbf{S}$, it is possible to control the properties of the wavelets and dual scaling functions that are built from the original scaling function $\bar{\Phi}^j$. Essentially, the new functions and operators are improved *lifted* versions of the old ones, and can have custom properties. Once the operator $\mathbf{S}$ is set, the lifting scheme keeps the bihortogonality

of the old filters to the new ones. Before getting to one practical example on how to build non-trivial wavelets from trivial ones, we will define the fast lifted wavelet transform and the dual lifting scheme.

### 5.4.2 Fast Lifted Wavelet Transform

Another benefit of the lifting scheme, is that it allows to write the wavelet transform just using the old filters and the $\mathbf{S}$ filter, without the need to explicitly derive the new ones. The forward lifted transform is

$$\begin{aligned} \mathbf{c}^j = \mathbf{A}^j \mathbf{c}^{j+1} &= \bar{\mathbf{A}}^j \mathbf{c}^{j+1} + \mathbf{S}^j \mathbf{d}^j \\ &= \bar{\mathbf{A}}^j \mathbf{c}^{j+1} + \mathbf{S}^j \bar{\mathbf{B}}^j \mathbf{c}^{j+1} \end{aligned} \tag{5.19}$$

The coarse signal $\mathbf{c}^j$ is calculated via the $\bar{\mathbf{A}}^j \mathbf{c}^{j+1}$ and then lifted with the details $\mathbf{d}^j$. Often in the literature this operation of lifting is called an *update*, and is denoted with the operator $U$. In the Eq. (5.19) the new filter $\mathbf{A}^j$ is never calculated explicitly.

The inverse lifted transform then becomes

$$\mathbf{c}^{j+1} = \mathbf{P}^j \mathbf{c}^j + \mathbf{Q}^j \mathbf{d}^j = \bar{\mathbf{P}}^j \left( \mathbf{c}^j - \mathbf{S}^j \mathbf{d}^j \right) + \bar{\mathbf{Q}}^j \mathbf{d}^j$$

It is possible to see that the operations in these transforms can be done inplace, meaning the only required storage is for the original signal $\mathbf{c}$ at the finest level. There is no need to calculate and store the full matrices, resulting in an easy implementation and a fast algorithm. We are not going in the details of the implementation since for our specific study and application the efficiency is not a primary requirement, and we actually compute the full matrices.

### 5.4.3 Dual Lifting Scheme

We have seen that via the lifting scheme it is possible to build an improved version of the starting matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{Q}}$, while the $\bar{\mathbf{P}}$ and $\bar{\mathbf{B}}$ remain unchanged.

It is possible to lift the dual step, and build a dual lifting scheme:

$$\begin{aligned}\mathbf{P}^j &= \bar{\mathbf{P}}^j + \bar{\mathbf{Q}}^j\tilde{\mathbf{S}}^j\\ \mathbf{A}^j &= \bar{\mathbf{A}}^j\\ \mathbf{Q}^j &= \bar{\mathbf{Q}}^j\\ \mathbf{B}^j &= \bar{\mathbf{B}}^j - \tilde{\mathbf{S}}^j\bar{\mathbf{A}}^j\end{aligned}$$

where the operators that are improved are the $\bar{\mathbf{P}}$ and $\bar{\mathbf{B}}$, while the other two remain unchanged. The dual operator $\tilde{\mathbf{S}}$ is often called *prediction* operator. Indicating explicitly the dimensions of the operators, as in Section 6.1.3,

$$\begin{aligned}\mathbf{P}^j_{M\times K} &= \bar{\mathbf{P}}^j_{M\times K} + \bar{\mathbf{Q}}^j_{M\times(M-K)}\tilde{\mathbf{S}}^j_{(M-K)\times K}\\ \mathbf{A}^j_{K\times M} &= \bar{\mathbf{A}}^j_{K\times M}\\ \mathbf{Q}^j_{M\times(M-K)} &= \bar{\mathbf{Q}}^j_{M\times(M-K)}\\ \mathbf{B}^j_{(M-K)\times M} &= \bar{\mathbf{B}}^j_{(M-K)\times M} - \tilde{\mathbf{S}}^j_{(M-K)\times K}\bar{\mathbf{A}}^j_{K\times M}\end{aligned}$$

In the following we will give an example of a common trivial wavelet, and build a lifted set of wavelets, the interpolating wavelet. This interpolating wavelet is going to be useful later in the dissertation.

### 5.4.4 The Lazy Wavelet

As a trivial set of filters to start the lifting process, it is possible to define two operators $\mathbf{E}$, $\mathbf{D}$ that essentially split the signal $\mathbf{c}^j$ into *even* and *odd* samples. These two operators are obviously orthogonal (as before we will assume that we are dealing with real operators)

$$\begin{bmatrix}\mathbf{E}\\ \mathbf{D}\end{bmatrix}\begin{bmatrix}\mathbf{E}^\top & \mathbf{D}^\top\end{bmatrix} = \begin{bmatrix}\mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{1}\end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix}\mathbf{E}^\top & \mathbf{D}^\top\end{bmatrix}\begin{bmatrix}\mathbf{E}\\ \mathbf{D}\end{bmatrix} = 1.$$

This means that the Lazy wavelet operators are exactly $\mathbf{E}$ and $\mathbf{D}$ only:

$$\mathbf{A}^j_{Lazy} = \mathbf{P}^{j\top}_{Lazy} = \mathbf{E} \qquad \text{and} \qquad \mathbf{B}^j_{Lazy} = \mathbf{Q}^{j\top}_{Lazy} = \mathbf{D}$$

The Lazy wavelet transform is a transform that splits and merges back the signal, without actually doing anything. This trivial set of filters is sufficient to define more interesting transforms, like the the interpolating wavelet transform.

### 5.4.5 Interpolating Wavelet Transform Built via the Lifting Scheme

First we have to note that any operator $\mathbf{W}$ can be split into two operators, one that acts on the even samples and one on the odd ones, like this:

$$\mathbf{W} = \mathbf{W}_e\mathbf{E} + \mathbf{W}_d\mathbf{D},$$

with

$$\mathbf{W}_e = \mathbf{W}\mathbf{E}^\top \qquad \text{and} \qquad \mathbf{W}_d = \mathbf{W}\mathbf{D}^\top. \tag{5.20}$$

By definition, an *interpolating filter* is a filter that satisfies this equation

$$\mathbf{P}^j_{int}\mathbf{E}^\top = 1.$$

The filter corresponding to the dual scaling function is then $\mathbf{A}^j_{int} = \mathbf{E}$. If we define $\tilde{\mathbf{S}}^j = \mathbf{P}^j_{int}\mathbf{D}^\top$, then from (5.20) any interpolating filter can be written as $\mathbf{P}^j_{int} = \mathbf{E} + \tilde{\mathbf{S}}^j\mathbf{D}$. This expression is equivalent to applying the dual lifting scheme to the Lazy wavelet, and so we can write the set of interpolating bihortogonal filters as

$$\begin{aligned}
\mathbf{P}^j_{int} &= \mathbf{E}^\top + \mathbf{D}^\top\tilde{\mathbf{S}}^j \\
\mathbf{A}^j_{int} &= \mathbf{E} \\
\mathbf{Q}^j_{int} &= \mathbf{D}^\top \\
\mathbf{B}^j_{int} &= \mathbf{D} - \tilde{\mathbf{S}}^j\mathbf{E}
\end{aligned} \tag{5.21}$$

Note that the $\mathbf{A}^j_{int}$ and $\mathbf{Q}^j_{int}$ are essentially Dirac deltas. These filters do not form a multiresolution analysis in $\mathcal{L}^2$, since the duals do not belong to $\mathcal{L}^2$. We

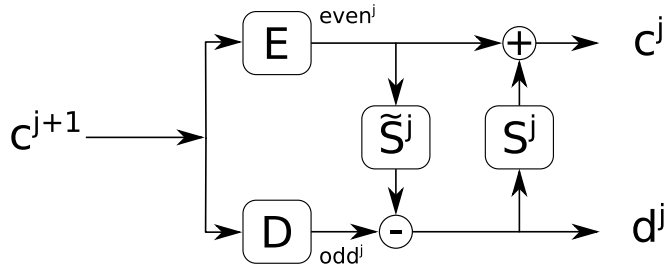



Figure 5.2: The interpolating wavelet transform is composed by three stages: the Lazy wavelet transform, the dual lifting and the normal lifting.

can apply the lifting scheme to the interpolating filters, so to improve the $\mathbf{A}^j_{int}$ and $\mathbf{Q}^j_{int}$, obtaining

$$\begin{aligned}
\mathbf{P}^j &= \mathbf{P}^j_{int} = \mathbf{E}^\top + \mathbf{D}^\top \tilde{\mathbf{S}}^j \\
\mathbf{A}^j &= \mathbf{A}^j_{int} + \mathbf{S}^j \mathbf{B}^j_{int} = (\mathbf{1} - \mathbf{S}^j \tilde{\mathbf{S}}^j)\mathbf{E} + \mathbf{S}^j \mathbf{D}^j \\
\mathbf{Q}^j &= \mathbf{Q}^j_{int} - \mathbf{P}^j_{int} \mathbf{S}^j = -\mathbf{E}^\top \mathbf{S}^j + \mathbf{D}^\top (\mathbf{1} - \tilde{\mathbf{S}}^j \mathbf{S}^j) \\
\mathbf{B}^j &= \mathbf{B}^j_{int} = \mathbf{D} - \tilde{\mathbf{S}}^j \mathbf{E}
\end{aligned} \tag{5.22}$$

The new filters we obtain are then result of applying first the Lazy wavelet transform (that performs the splitting), then the dual lifting step and finally the regular lifting. Quite often in literature the union of these three operations it is identified with the *lifting scheme*, even if the lifting is actually only one stage. In Figure 5.2 we illustrate the flow of these operations. From the scheme in Figure 5.2 we can also manually derive the two filters $\mathbf{A}^j$ and $\mathbf{B}^j$ just following the arrows. The filter $\mathbf{A}^j$, for example, connects the $\mathbf{c}^{j+1}$ with the $\mathbf{c}^j$, and there are three paths that connect the two: from $\mathbf{c}^{j+1}$ via $\mathbf{E}$ directly to $\mathbf{c}^j$, from $\mathbf{c}^{j+1}$ via $\mathbf{D}$ and $\mathbf{S}^j$ and the last one goes through $\mathbf{E}$, $\tilde{\mathbf{S}}^j$, back via $\mathbf{S}^j$ and to $\mathbf{c}^j$. And we get $\mathbf{A}^j = \mathbf{E} - \tilde{\mathbf{S}}^j \mathbf{E} + \mathbf{S}^j \mathbf{D}$. Similarly, to build the filter $\mathbf{B}^j$ that connects the $\mathbf{c}^{j+1}$ with the $\mathbf{d}^j$ there are two paths: one that goes directly through $\mathbf{D}$, and one that gets to $\mathbf{d}^j$ via $\mathbf{E}$ and $\tilde{\mathbf{S}}^j$. Giving exactly $\mathbf{B}^j = \mathbf{D} - \tilde{\mathbf{S}}^j \mathbf{E}$.

Figure 5.3 shows the scheme for the complete interpolating wavelet transform with decomposition and reconstruction. It is possible to notice that the

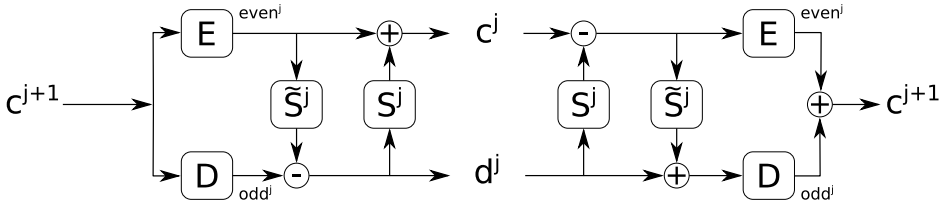



Figure 5.3: The complete interpolating wavelet transform with decomposition and reconstruction. The reconstruction is made by the same steps as decomposition, but performed backwards.

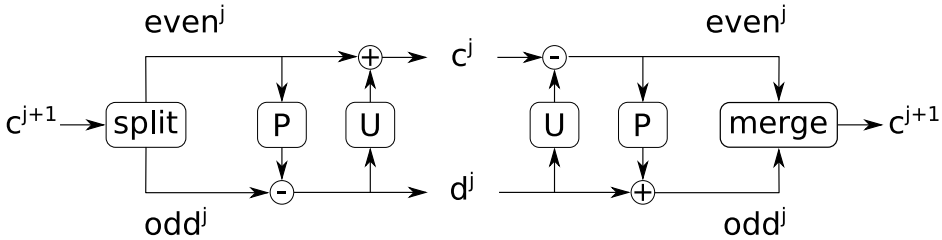



Figure 5.4: Alternative writing of the complete interpolating wavelet transform scheme. This form of writing is probably the most common in literature.

reconstruction stage is made by exactly the same steps as the decomposition, but performed backwards.

In Figure 5.4 we show an alternative way of depicting the lifting scheme that is quite common in literature. The operator $\tilde{\mathbf{S}}^j$ is called *prediction* operator $P$, while the $\mathbf{S}^j$ is called *update* $U$. Here we preferred to keep the operator $\mathbf{P}$ as the upsampling operator from $\mathbf{c}^j$ to $\mathbf{c}^{j+1}$ and avoid confusion with different typesets of $P$. In this same figure it is possible to notice that the operators $\mathbf{E}$ and $\mathbf{D}$ are replaced by two boxes called *split* and *merge*. Often they are depicted with boxes containing a $[\downarrow 2]$ for the split stage (meaning downsample by a factor of 2) and a $[\uparrow 2]$ for the merge (meaning upsample by a factor of 2). To give a broader picture, since the wavelet literature its very diverse in origin and scope, we wanted to collect here some different notations used in other frameworks. Nevertheless the meaning is exactly the same.

We have described the lifting scheme as a way to build improved operators starting from trivial ones. Another way to understand the lifting scheme is

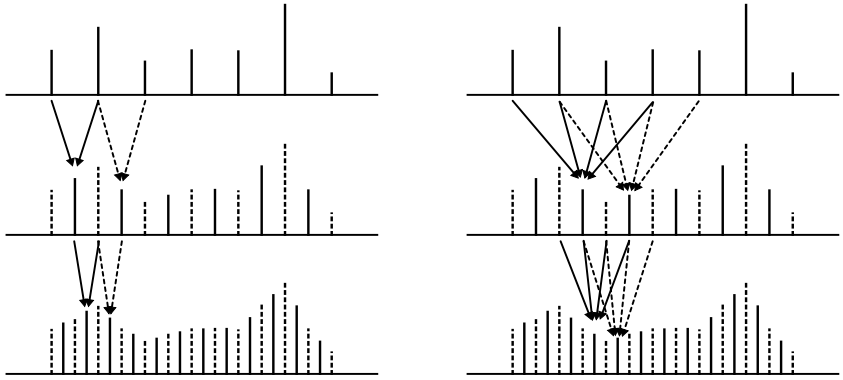


Figure 5.5: Graphic representation of linear (left) and cubic (right) prediction operators in the reconstruction stage. The idea for the graphical illustration is taken from [Sweldens and Schröder, 1995].

by interpreting the $\tilde{\mathbf{S}}^j$ and $\mathbf{S}^j$ as prediction and update operators. Assuming that the original signal has some sort of local correlation, once the signal is split into odd and even subsets, these two signals are highly correlated. This means that given one, it should be possible to predict the other with a certain accuracy. And this is what the prediction operator does, e.g. getting an odd sample using its even neighbour(s). Intuitive examples of prediction operator are the polinomial prediction operators, e.g. linear, quadratic or cubic ones, where 2, 3 or four even neighbours respectively are used to predict one odd sample. A graphic illustration is reported in Figure 5.5. The update operator is designed so to preserve the overall average of the signal. The idea is that the coarser signals $\mathbf{c}^j$ have the same average value of the original signal, and going down to last possible level $\mathbf{c}^0$ this will capture its constant offset (or average). This is equivalent to ask for zero average details $\mathbf{d}^j$.

Note: recall the intuitive definition given in Section 5.4.4 for odd and even operators, that essentially split a signal in two signals based on the sample index. When moving to spaces that are more complex than the line or the plane, e.g. generic meshes, this simple definition can be generalized and the lifting scheme can still be applied in the very same way. As an example, we will see now how the lifting scheme generalizes on a spheric mesh and how we can

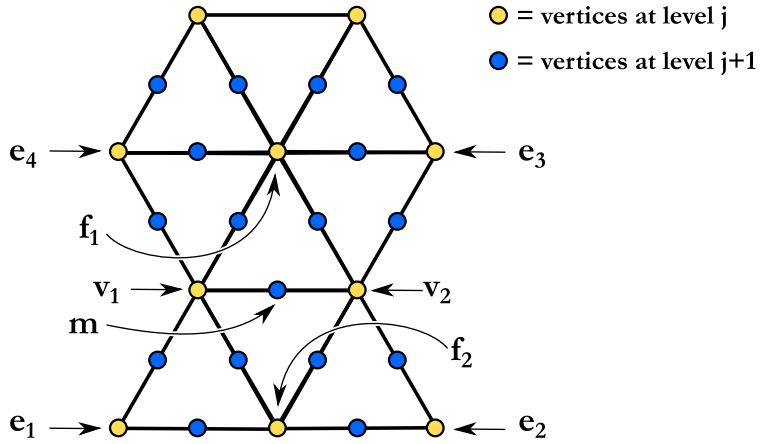



Figure 5.6: Mesh neighbours. Given $m$ a point on the $j+1$ mesh, we define three levels of neighbourhood: the closest ones $\{v_1, v_2\}$, then $\{f_1, f_2\}$ and $\{e_1, e_2, e_3, e_4\}$.

build interpolating wavelets on this sphere.

## 5.5 Second Generation Spherical Wavelets via the Lifting Scheme

To build wavelets on spaces other than the line, we need a data structure with hierarchical subdivision and a (re)definition of odd and even samples (or vertices). In Section 5.3 we defined already a subdivision mesh based on the interative Loop subdivision of an octahedron. In Figure 5.6 we show an example of a mesh obtained via Loop subdivision. The yellow dots represent the vertices of the mesh at level $j$, while the blue dots (plus the yellow ones) represent the vertices of the mesh at level $j+1$. If we take into consideration the point $m$ of the $j+1$ level, that would be the odd vertex in the previous dissertation, then its neighbours are defined at different distances. The vertices $\{v_1, v_2\}$ represent the even vertices, and are at equal distance from $m$. With two points we can already build the linear interpolating wavelet transform. If we want to build interpolating wavelets on this mesh with higher predictivity, we have to define other (further) neighbours searching for close vertices at level $j$. The

next closest points are the two $\{f_1, f_2\}$, and the successive ones are the four $\{e_1, e_2, e_3, e_4\}$.

Defined these three sets of neighbours, to operatively build the wavelets we have to define the lifting operators $\tilde{\mathbf{S}}$ and/or the $\mathbf{S}$. In the paper from [Schröder and Sweldens, 1995] the dual lifting operators to generate linear, quadratic and butterfly wavelet filters are defined. Here we are interested in the linear interpolating wavelet transform, which is not much different to the real line case, since the prediction operator uses only the immediate neighbours. Recalling the expressions in (5.21), the dual lifting step (or the prediction step) for the interpolating wavelets is for the analysis (lifting of $\mathbf{B}_{int}$ and $\mathbf{P}_{int}$)

$$\begin{aligned} \mathbf{c}^{\text{dual}}_{j,k} &= \mathbf{c}_{j+1,k} && \text{even} \quad \mathbf{A}_{int} \\ \mathbf{d}^{\text{dual}}_{j,m} &= \mathbf{c}_{j+1,m} - 1/2(\mathbf{c}_{j+1,v_1} + \mathbf{c}_{j+1,v_2}) && \text{odd} \quad \mathbf{B}_{int} \end{aligned} \tag{5.23}$$

and for the synthesis

$$\begin{aligned} \mathbf{c}_{j+1,k} &= \mathbf{c}^{\text{dual}}_{j,k} && \text{even} \quad \mathbf{Q}_{int} \\ \mathbf{c}_{j+1,m} &= \mathbf{d}^{\text{dual}}_{j,m} + 1/2(\mathbf{c}^{\text{dual}}_{j,v_1} + \mathbf{c}^{\text{dual}}_{j,v_2}) && \text{odd} \quad \mathbf{P}_{int} \end{aligned} \tag{5.24}$$

The dual lifting weights are then $\tilde{s}_{j,v_1,m} = \tilde{s}_{j,v_2,m} = 1/2$.

This construction can be lifted and the weights for the lifting step or update step are chosen so that the wavelet has zero integral

$$\mathbf{s}_{j,k,m} = I_{j+1,m}/2I_{j,k}, \text{ with } I_{j,k} = \int_{\mathbb{S}^2} \psi_{j,k} d\omega$$

and the integrals $I_{j,k}$ can be approximated at the finest level, and calulated resursively at coarser levels using the refinement relations. The update step updates the coarse coefficients via the details, as seen in (5.18). Explicitly, for the analysis (lifting of $\mathbf{A}_{int}$)

$$\begin{aligned} \mathbf{c}^{\text{lift}}_{j,v_1} &= \mathbf{c}^{\text{dual}}_{j,v_1} + \mathbf{s}_{j,v_1,m}\mathbf{d}^{\text{dual}}_{j,m} \\ \mathbf{c}^{\text{lift}}_{j,v_2} &= \mathbf{c}^{\text{dual}}_{j,v_2} + \mathbf{s}_{j,v_2,m}\mathbf{d}^{\text{dual}}_{j,m} \end{aligned} \tag{5.25}$$

and for during synthesis the analysis step is essentially undone (lifting of $\mathbf{Q}_{int}$)

$$\begin{aligned} \mathbf{c}^{\text{dual}}_{j,v_1} &= \mathbf{c}^{\text{lift}}_{j,v_1} - \mathbf{s}_{j,v_1,m}\mathbf{d}^{\text{dual}}_{j,m} \\ \mathbf{c}^{\text{dual}}_{j,v_2} &= \mathbf{c}^{\text{lift}}_{j,v_2} - \mathbf{s}_{j,v_2,m}\mathbf{d}^{\text{dual}}_{j,m}. \end{aligned} \tag{5.26}$$

The signal flow is then: apply the prediction step (5.23), then the update (5.25) for the analysis stage, and then undo the update (5.26) and undo the prediction (5.24) for the synthesis.

The resulting interpolating wavelets and scaling functions, for the mesh described in Section 5.3, are depicted in Figures 5.7, 5.8, 5.9 and 5.10. Note that in this particular space, the sphere, because there are no boundaries the wavelets are all identical at the same level. For this reason we show only one filter per level. In these figures it is possible to realize that higher level filters are actually shrunk versions of lower levels' ones.

Note: in the lifting scheme we never build explicitly the scaling functions or wavelets, but we can get them by running a delta into the graph and running it ad infinitum, as described in (5.17).

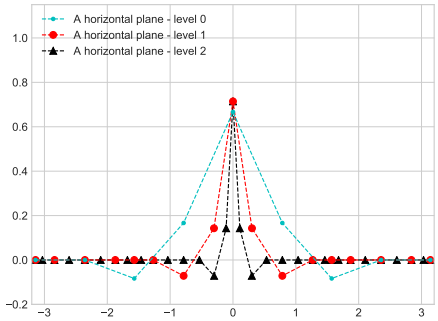



Figure 5.7: Dual scaling filter at levels 0, 1, 2 (given by one row of $\mathbf{A}^{1,2,3}$).

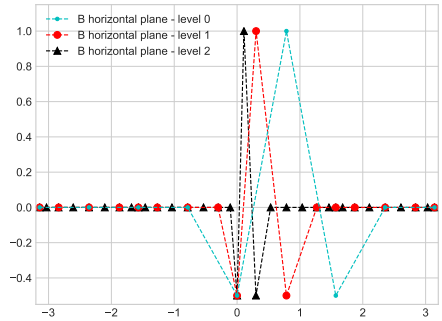



Figure 5.8: Dual wavelet filter at levels 0, 1, 2 (given by one row of $\mathbf{B}^{1,2,3}$).

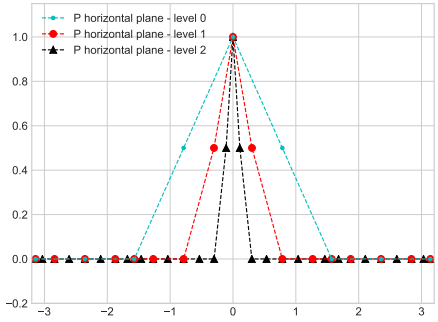



Figure 5.9: Scaling filter at levels 0, 1, 2 (given by one column of $\mathbf{P}^{1,2,3}$).

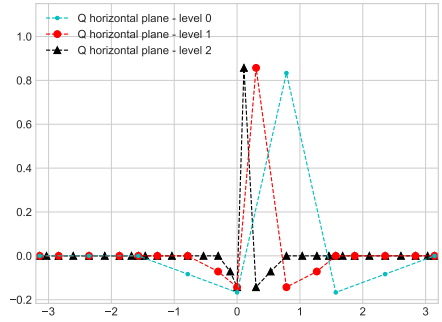



Figure 5.10: Wavelet filter at levels 0, 1, 2 (given by one column of $\mathbf{Q}^{1,2,3}$).

# Chapter 6

# WAVELET-BASED SPHERICAL AUDIO FRAMEWORK

In this Chapter we will describe the full audio chain for a wavelet based audio format. Section 6.1 describes the multiresolution scheme over a subdivision mesh. Section 6.2 depicts the strategy for the encoding of an audio source over the subdivision mesh. Up to this point, the setup is completely general, and can be used to generate a wide diversity of formats. For this reason we call this audio encoding scheme a *framework* for wavelet audio formats. In Section 6.3 we particularize this wavelet format to the spherical domain. Section 6.4 is about the decoding of the new spherical wavelet format.

## 6.1 Wavelet Format

In this section we will describe the basics of wavelet multiresoution analysis, with emphasis on the practical aspects that are relevant for an audio encoding/decoding chain.

### 6.1.1 Decomposition or Spatial Downsampling

Being $\mathbb{M}$ some mesh in $\mathbb{R}^3$, obtained with some subdivision scheme, then the set of data $\mathbf{f} = (f_1 \ \cdots \ f_N)^T$ defined over the finest level of this mesh, is called

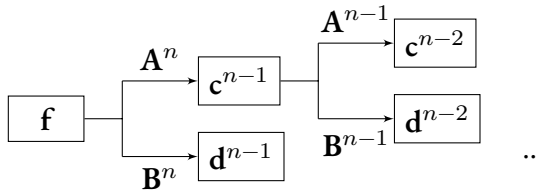



Figure 6.1: Scheme of signal decomposition and encoding to wavelet space.

*fine data*. The process of downsampling decomposes the fine data $\mathbf{f}$ into two signals (sets of data), a *coarse* approximation $\mathbf{c}$ and an additional information called *details* $\mathbf{d}$. The decomposition is defined then as:[1]

$$\mathbf{c} = \mathbf{A}\mathbf{f},$$
$$\mathbf{d} = \mathbf{B}\mathbf{f},$$

where $\mathbf{A}$ and $\mathbf{B}$ are the *decomposition* or *analysis filters* introduced in Section 5.2.

The filters $\mathbf{A}$ and $\mathbf{B}$ connect levels: from the finest level $n$ to a coarser level $n-1$. There are as many decomposition filters, or encoding matrices, as mesh levels minus one. The signal $\mathbf{c}$ represents a spatially low-passed and downsampled version of $\mathbf{f}$.

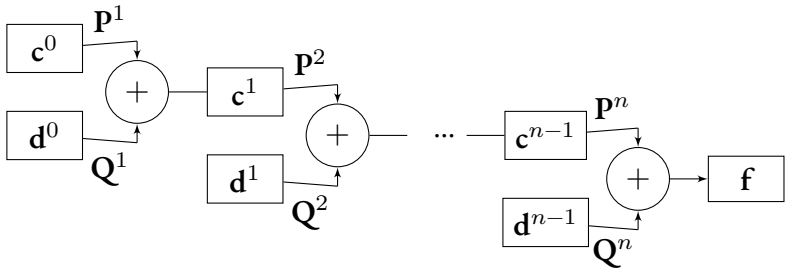



Figure 6.2: Scheme of reconstruction of the original signal from the wavelet space.

[1] For the definition of scaling and wavelet filters, we use an adaptation of the the notation in [Olsen et al., 2007].

### 6.1.2 Reconstruction or Spatial Upsampling

The upsampling process increases the spatial resolution of the coarse data **c** to the fine data **f**, and if the details **d** are available, then the reconstruction process will give back the original fine data:

$$\mathbf{f} = \mathbf{P}\mathbf{c} + \mathbf{Q}\mathbf{d}.$$

where **P** and **Q** are the *reconstruction* or *synthesis filters* introduced in Section 5.2. The filters **P** and **Q** connect levels: from the coarser level $n-1$ to the finest level $n$. There are as many reconstruction filters, or decoding matrices, as mesh levels minus one.

### 6.1.3 Wavelet Transform

Consider the finest subdivision level $n$, consisting of $N$ points, and the next coarser level $n-1$, consisting of $M$ points, with $N > M$. Consider a data vector $\mathbf{f} = [f_1, \dots, f_N]^T$, defined at the finest level $n$. The subdivision process would proceed as follows:

$$\begin{aligned} \mathbf{c}^{n-1}_{M\times 1} &= \mathbf{A}^n_{M\times N}\mathbf{f}_{N\times 1}, \\ \mathbf{d}^{n-1}_{(N-M)\times 1} &= \mathbf{B}^n_{(N-M)\times N}\mathbf{f}_{N\times 1}, \end{aligned}$$

where the subindices indicate the dimension of each matrix. The sum of elements in **c** and **d** is $M + (N-M) = N$. Now if the even coarser subdivision level $n-2$ has $K$ points, the decomposition can continue:

$$\begin{aligned} \mathbf{c}^{n-2}_{K\times 1} &= \mathbf{A}^{n-1}_{K\times M}\mathbf{c}^{n-1}_{M\times 1}, \\ \mathbf{d}^{n-2}_{(M-K)\times 1} &= \mathbf{B}^{n-1}_{(M-K)\times M}\mathbf{c}^{n-1}_{M\times 1}, \end{aligned}$$

and so on. See Figure 6.1 for a representation.

If the decomposition is followed up to the coarsest level available (*level 0*), there will be a list of $n-1$ detail signals or wavelet coefficients, $\mathbf{d}^0, \dots, \mathbf{d}^{n-1}$, and one last coarse signal or scaling function coefficients $\mathbf{c}^0$; the representation

$\{\mathbf{c}^0, \mathbf{d}^0, \ldots, \mathbf{d}^{n-1}\}$ constitutes the *wavelet transform*. In compact form, the wavelet transform can be computed recursively as:

$$\begin{aligned}\mathbf{c}^{j-1} &= \mathbf{A}^j \mathbf{c}^j, \\ \mathbf{d}^{j-1} &= \mathbf{B}^j \mathbf{c}^j,\end{aligned}$$

with $\mathbf{c}^n = \mathbf{f}$ and $j = n, \ldots, 1$.

To perform the inverse wavelet transform, namely, to reconstruct the original signal, the procedure is recursive but this time starting from the coarsest *level 0* and going to the finest *level* $n$:

$$\begin{aligned}\mathbf{c}^1_{J\times 1} &= \mathbf{Q}^1_{J\times(J-L)}\mathbf{d}^0_{(J-L)\times 1} + \mathbf{P}^1_{J\times L}\mathbf{c}^0_{L\times 1} \\ \mathbf{c}^2_{K\times 1} &= \mathbf{Q}^2_{K\times(K-J)}\mathbf{d}^1_{(K-J)\times 1} + \mathbf{P}^2_{K\times J}\mathbf{c}^1_{J\times 1},\end{aligned}$$

see Figure 6.2 for a diagram. In compact form, the inverse wavelet transform can be represented recursively as:

$$\mathbf{c}^k = \mathbf{P}^k \mathbf{c}^{k-1} + \mathbf{Q}^k \mathbf{d}^{k-1}, \tag{6.3}$$

with $k = 1, \ldots, n$.

The filters $\mathbf{A}^j$, $\mathbf{B}^j$, $\mathbf{P}^j$ and $\mathbf{Q}^j$ are the building blocks of the proposed spatial audio format. They are not arbitrary: to define a wavelet framework they need to follow relations (5.14) and (5.16). Several methods to build these filters are available in literature. In Section 5.4 we describe a method based on the Lifting Scheme in Section 5.5, and in Chapter 7 our optimization scheme. In the following we will assume that some filters with appropriate characteristics are available.

A note of caution: the matrices $\mathbf{A}^j$ and $\mathbf{B}^j$ connect the level $j$ with the level $j-1$, while the matrices $\mathbf{P}^j$ and $\mathbf{Q}^j$ connect the level $j-1$ with the $j$. The index of the matrices is the same, but the dimension of their output signal is different.

## 6.2 Audio Source Encoding

### 6.2.1 First Step: Source Interpolation

In spatial audio, any source can have an arbitrary position in spherical space, with continuous azimuth and elevation coordinates $(\theta, \phi)$. This point source needs to be represented on the finest mesh, which is defined only over a discrete set of points. The first step of the encoding process is the interpolation of the point source to the finest mesh. The most natural interpolation for triangular meshes is the tri-linear interpolation over the three vertices of the triangle. In spatial audio this tri-linear interpolation is the basis of Vector-Base Amplitude Panning [Pulkki, 1997] (VBAP). In this thesis we use a tri-linear (or VBAP-like) interpolation to represent the point source in the subdivision mesh (other choices are also possible).

### 6.2.2 Second Step: Wavelet Encoding

As a result of the interpolation, a set of coefficients $\mathbf{f} = (f_1 \ \cdots \ f_N)^T$ will be available at the finest mesh (order $n$). If the source is a point source, at most three of these coefficients will be non-zero. Second step is to apply the wavelet transform, recursively downsampling the subdivision mesh, by repeated application of the decomposition filters **A** and **B**, as explained in Section 6.1 and described in Figure 6.1. The result of the wavelet transform will be the set of signals $\{\mathbf{c}^0, \mathbf{d}^0, \dots, \mathbf{d}^{n-1}\}$, having the same total dimensionality $N$ as the original.

### 6.2.3 Third Step: Wavelet Truncation

At this stage, one of the most common techniques in different fields is zeroing all the details coefficients smaller than some fixed threshold in order to achieve compression. In spatial audio, the goal is also to transmit a low-dimensional field, but to ensure a smooth and consistent playback experience, and in analogy to Ambisonics, we will follow a different path by limiting the decomposition up to a given order.

Therefore, the third step in the encoding is truncating the decomposition to order $\ell$, with $0 \leq \ell < n$, which amounts to zeroing the detail coefficients with order equal or greater than $\ell$. Namely, the truncated decomposition at order $l$ be $\{\mathbf{c}^0, \mathbf{d}^0, \ldots, \mathbf{d}^{\ell-1}, \mathbf{0}^\ell, \ldots, \mathbf{0}^{n-1}\}$, or, more simply, $\{\mathbf{c}^0, \mathbf{d}^0, \ldots, \mathbf{d}^{\ell-1}\}$. (In the case $\ell = 0$ all wavalet coefficients will be zero, with only the coarser scaling coefficient remaining.)

## 6.3 Spherical Wavelet Format

A *Spherical Wavelet Format* (SWF) is defined to be each one of the spherical audio encodings determined by:

1. A recursive subdivision mesh over the sphere, ranging from the coarsest level 0 to the finest level $n$.

2. A set of filters $\{\mathbf{A}^j, \mathbf{B}^j, \mathbf{P}^j, \mathbf{Q}^j | j \in [1, n]\}$, defining a wavelet space, and verifying the set of equations (5.14) and (5.16).

3. A truncation level $\ell \in [0, n]$, defining the order of the wavelet decomposition.

The signals in SWF can be represented in two alternative equivalent ways: either as the scaling function at the coarser level plus a set of successively finer wavelet details, $\{\mathbf{c}^0, \mathbf{d}^0, \ldots, \mathbf{d}^{\ell-1}\}$, or as the scaling functions at the truncation level $\ell$: $\{\mathbf{c}^\ell\}$. In this second representation only the downsampling $\mathbf{A}^j$ and upsampling $\mathbf{P}^j$ filters are strictly needed.

There are many different ways to generate the filters defining the wavelet space. One method is the Lifting Scheme, described in Section 5.4. Other methods build on the lifting scheme, generating optimized filter for specific applications e.g. [Kammoun et al., 2012]. In Chapter 7 we describe our audio-tailored method.

Another possibility is given by the application of VBAP as the interpolator between mesh levels. VBAP implicitly defines a set of downsampling, $\mathbf{A}^j$ and upsampling $\mathbf{P}^j$ filters for any subdivision mesh. This filters can be computed

by considering a given mesh at level $j$ as a set of sound sources, and rendering those meshes to the finer level ($\mathbf{P}^j$ is generated) or coarser level ($\mathbf{A}^j$ is generated). In this case the filters created via VBAP have a maximum length of 3 points for $\mathbf{A}^j$, because VBAP activates at most three neighbouring points of the mesh. In the case of $\mathbf{P}^j$ VBAP generates a trivial filter, which is a block matrix with an identity on the first block and zeros elsewhere (since the points of level $j$ are contained in the level $j+1$ mesh), so there is no effective upsampling. These VBAP-inspired wavelets are close in spirit to the interpolating wavelets [Schröder and Sweldens, 1995], with the difference of having a different set of neighbours and with the dual and direct spaces swapped. We will call this version of SWF based on VBAP: VBAP-SWF.

## 6.4 Spherical Wavelet Format Decoding

Let us consider a SFW encoding of an arbitrary sound source distribution. If this wavelet encoding has not been truncated, a trivial decoding can be generated by inverting the wavelet transform, making repeated use of Eq. (6.3). This will lead to the original source $\mathbf{c}^n = \mathbf{f}$. By associating a loudspeaker to each point of the finer mesh, the values of $\mathbf{f}$ can be interpreted as the loudspeaker signals. This is represented on the first four blocks on the left column of Figure 6.3. Essentially, this trivial decoding reproduces the VBAP decoding of the original sound source to the finer mesh.

A more interesting case happens when the wavelet encoding has been truncated to order $\ell < n$: the encoding can be partially inverted to order $\ell$, again by repeated use of Eq. (6.3). The truncated signal can be represented by the scaling coefficients at order $\ell$, $\mathbf{c}^\ell$. If the nodes of the $\ell$-th mesh are interpreted as speakers, then the values of $\mathbf{c}^\ell$ can be considered as the speaker feeds corresponding to a decoding to a regular layout with one loudspeaker in each one of the vertices of the mesh at $\ell$-th level.

Actually, the signal at level $\ell$ can be upsampled by applying repeatedly the reconstruction filter $\mathbf{P}$, ignoring the details beyond level $\ell$:

$$\tilde{\mathbf{c}}^{\ell+1} = \mathbf{P}^{\ell+1}\mathbf{c}^\ell,$$

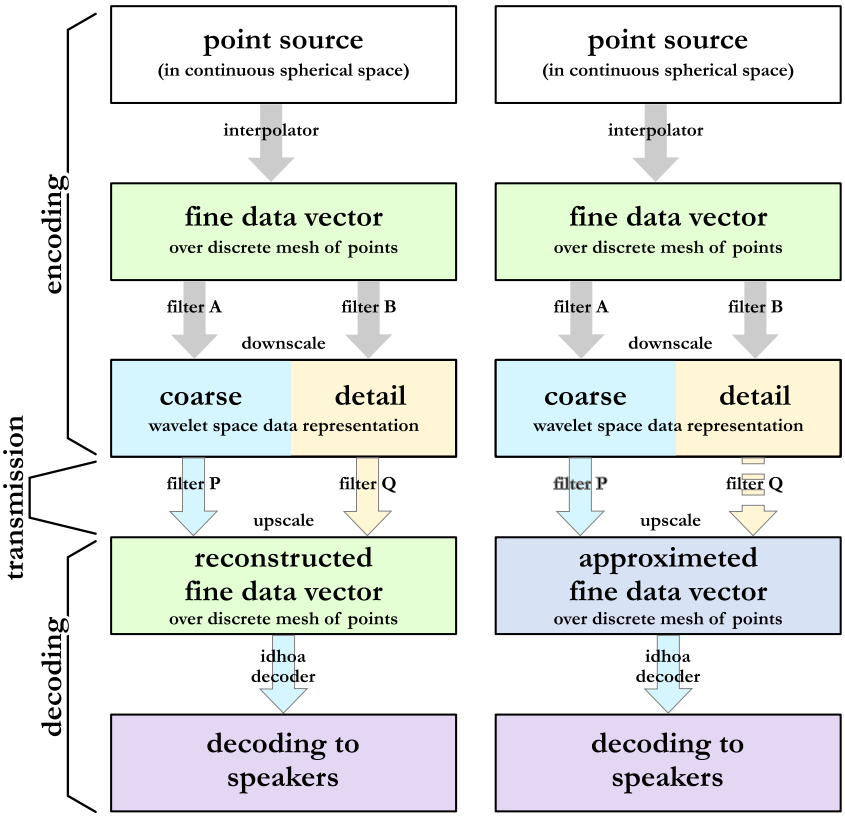



Figure 6.3: Encoding, transmission and decoding of a SWF, without wavelet truncation (on the left), and with wavelet truncation (on the right).

where the tilde indicates that the reconstruction comes from a truncated representation. Ultimately, the signal can be reconstructed to the finest level. The resulting signal $\tilde{\mathbf{f}} = \tilde{\mathbf{c}}^n$, corresponds to a spatial low-pass version of the original signal $\mathbf{f}$.

This way, by interpreting the coefficients $\tilde{\mathbf{c}}^k$ as loudspeaker signals associated to the vertices of the meshes at level $k$, decodings to the different layouts corresponding to the mesh geometry at different levels are generated, from the coarse level $\ell$ to the finest level $n$. This is represented on the first four blocks on the right column of Figure 6.3. This decoding procedure is the wavelet equivalent to the basic decoding of Ambisonics to a regular layout. It is to be noted however that in wavelets, differently to Ambisonics, there is no guarantee that the pressure, the acoustic velocity or any other relevant acoustical or psychoacoustical parameter are correctly reconstructed at the origin, unless the wavelet family has been designed in some special way.

Here we propose to optimize the matrix responsible for the decoding to speakers leveraging some acoustical and psychoacoustical observables, as we did for Ambisonics with IDHOA (see Chapter 3).

### 6.4.1 Decoding of SWF to an Irregular Loudspeaker Layout

The goal is, given a loudspeaker layout and a given SWF signal, to generate the speaker signals $\mathbf{s}$ as a linear combination of the scaling function coefficients at the truncation order $\ell$:

$$\mathbf{s} = \mathbf{D}^{\ell+1}\mathbf{c}^{\ell}$$

This is represented on the bottom blocks of Figure 6.3. The decoding matrix has as many rows as speakers and as many columns as channels in the SWF. The scaling function channels at level $\ell$, $\mathbf{c}^{\ell}$, can be computed out of the channels in the wavelet transform representation, $\{\mathbf{c}^0, \mathbf{d}^0, \dots, \mathbf{d}^{\ell-1}\}$, by applying repeatedly Eq. (6.3). Actually, the decoding matrix can be also computed from a upsampled version of the coarse channels:

$$\mathbf{s} = \mathbf{D}^{j+1}\tilde{\mathbf{c}}^{j}, \quad \ell < j \leq n.$$

The decoded $\mathbf{s}$ produced via $\mathbf{D}^{\ell+1}$ and $\mathbf{D}^{j+1}$ should be equivalent, but in practice, since the different decoding matrices are obtained a via separate non-linear optimization processes, there might be differences.

We have developed a numerical optimization method to find the optimal decoding of a given SWF, based on IDHOA. With respect to the implementation described in Chapter 3, IDHOA has been adapted to accept any input in any format, as long as a sound source can be encoded in a sufficient number of points on the sphere of the sphere. With 'sufficient' we mean that the number of points where the sound source can be encoded has to be greater than the number of speakers of the destination layout. The same perceptual criteria used for Ambisonics are still valid for Wavelets decoding: pressure, energy, radial velocity, radial intensity, while the transverse velocity and transverse intensity are minimized. A quadratic term is responsible for penalizing negative gains. This way IDHOA can now produce decoding matrices for both Ambisonics and SWF.

Schematically:

1. *Initialization:* Operations that are performed only once when the algorithm is launched.

2. Given the loudspeakers' layout, calculate $\mathbf{D}_{\text{init}}$. In this case we might not have a hint on the decoding matrix, like the projection matrix in Ambisonics, so we can initialize $\mathbf{D}_{\text{init}}$ completely random.

3. Calculate the various physical variables that constitute the cost function: $p$, $E$, $\mathbf{v}$, $\mathbf{I}$. In the case of SWF, the set of $n$ points where is actually possible to evaluate these variables coincides with the set of points of the finest mesh at which the wavelets are available. (The set of sampling points is not arbitrary anymore like for Ambisonics, where the functions defining the format are continuous and it is possible to calculate the encoding matrix at any point in space.) Calculate the objective function, which is $f = f(\mathbf{D}_{\text{init}})$.

4. *Fix constraints (optional):* constrain some parameters to have a fixed value (e.g. lock to zero).

5. *Minimization stage:* Call to the external minimization algorithm, passing $\mathbf{D}_{\text{init}}$ and $f$. When the minimization algorithm terminates, it returns a $\widetilde{\mathbf{D}}$.

## 6.5 Summary

In this Chapter we have described a method for spatial audio encoding and decoding leveraging the multiresolution paradigm. The encoding and decoding filters are built directly on the multiresolution mesh (in Section 5.5 and Chapter 7 we show two different methods to actually build them). The final step of decoding to speakers is demanded to IDHOA, leveraging the same observables used for Ambisonics decoding.

# Chapter 7

# BUILDING SECOND GENERATION SPHERICAL WAVELETS VIA NUMERICAL OPTIMIZATION

## 7.1 Motivation

The application of wavelets in spatial audio is very different from the other fields such as computer graphics or image compression. On the one hand, the number of coefficients is very limited, e.g. between 4 and 20, while typically for analysis or compression the number of coefficients can be of the order of thousands; this fact impacts the maximum length of the filters and their shape. On the other hand, the tuning of the analysis and synthesis filters is not targeted to the minimization of some reconstruction error, but to specific characteristics: pressure preservation, smooth filtering, or limiting the negative components of the filters (which correspond to out-of-phase contributions).

In the conclusions of [Dremin et al., 2001], they say:

> "[...] the wavelet applications in various fields are numerous and give nowadays very fruitful outcome. [...] The potentialities of

> wavelets are still not used at their full strength. However one should not cherish vain hopes that this machinery works automatically in all situations by using its internal logic and does not require any intuition. According to [Wickerhauser, 1994], "no 'universal algorithm' is appropriate for the extreme diversity of the situations encountered". Actually it needs a lot of experience in choosing the proper wavelets, in suitable formulation of the problem under investigation, in considering most important scales and characteristics describing the analyzed signal, in the proper choice of the algorithms (i.e., the methodology) used, in studying the intervening singularities, in avoiding possible instabilities etc. By this remark we would not like to prevent newcomers from entering the field but, quite to the contrary, to attract those who are not afraid of hard but exciting research and experience."

We found that the standard lifting scheme, while it makes it very easy to tune the characteristics of the synthesis operator, makes the tuning of the analysis operator much more challenging, often leading to very non-smooth constructions. Looking at Eq. (5.22) it is apparent that in the construction of $\mathbf{A}^j$ and $\mathbf{Q}^j$, both lifting and dual lifting steps get mixed together. There is no way of constructing, for example, a $\mathbf{P}^j$ without affecting the $\mathbf{A}^j$ (and vice versa).

## 7.2 Numerical Optimization

We instead designed a brute force approach based on numerical optimization which aims at optimizing simultaneously both $\mathbf{A}^j$ and $\mathbf{P}^j$ operators, retaining the idea of locality of the wavelets and the symmetries of the multiresolution.

The unknowns of the problem are all the four operators: $\mathbf{A}^j$, $\mathbf{B}^j$, $\mathbf{P}^j$ and $\mathbf{Q}^j$. Since our main interest is on the (scaling function) operators $\mathbf{A}$ and $\mathbf{P}$, the optimization problem has been split into two stages: *stage 1* with $\mathbf{A}$, $\mathbf{P}$ as unknowns, and *stage 2* with $\mathbf{B}$, $\mathbf{Q}$ unknowns. All the unknowns grow quadratically with the number of points in the mesh, and so their constraints

(i.e. biorthogonality relations, Eq. (5.15)). For this reason we exploit the regularity of the mesh, that reflects in symmetries in the target matrices, to reduce the dimensionality of the problem (see Appendix B). The structure of the mesh also suggests some relations of neighbourhood, as we discussed in Section 5.5 with the help of Figure 5.6.

For the first stage, the optimization of scaling function operators, the cost function is made of 4 terms:

$$C = \alpha_\Lambda C_\Lambda + \alpha_{p1} C_{p1} + \alpha_{p2} C_{p2} + \alpha_{\text{neg}} C_{\text{neg}},$$

which are optimized with respect to operators $\mathbf{A}^j$ and $\mathbf{P}^j$. The first term is related to the *shape* of the low-pass filtering induced by $\mathbf{A}$ and $\mathbf{P}$. Being $\Lambda$ the desired target for the subsequent application of $\mathbf{A}$ and $\mathbf{P}$, the associated cost term is:

$$C_\Lambda = \frac{1}{N_k N_m} \sum_{k,m} \left[ \sum_l p^j_{kl} a^j_{lm} - \Lambda_{km} \right]^2,$$

where $a^j_{lm} \in \mathbf{A}^j$ and $p^j_{kl} \in \mathbf{P}^j$ and $N_k$ and $N_m$ indicate the number of elements in each one of the terms in the sum. The second term asks for *pressure preservation* during decomposition, and is:

$$C_{p1} = \frac{1}{N_m} \sum_m \left[ \sum_l a^j_{lm} - 1 \right]^2,$$

The third term asks for *pressure preservation* across decomposition and reconstruction, also among different levels, and is:

$$C_{p2} = \sum_{j'=0}^{j} \frac{1}{N_m} \sum_m \left[ \sum_k \left( \mathbf{P}^j \cdots \mathbf{P}^{j'} \mathbf{A}^{j'} \cdots \mathbf{A}^j \right)_{km} - 1 \right]^2.$$

The fourth and last term asks for positive panning laws, penalizing negative

coefficients, and is a condition on the matrices alone:

$$C_{\text{neg}} = \frac{1}{N_l N_m} \sum_{jk} (a^j_{lm})^2 \theta(-a_{lm}) + \frac{1}{N_k N_l} \sum_{kl} (p^j_{kl})^2 \theta(-p^j_{kl}).$$

Besides, there is an orthonormality constraint

$$\mathbf{A}^j \mathbf{P}^j = \mathbf{1}.$$

The optimization is then performed level by level, from level 0 to level $\ell$, leaving all previous coarser levels frozen in each subsequent step.

For the second stage, once **A** and **P** are set, the wavelet operators **B** and **Q** can be obtained almost algebraically, from the requirement $\mathbf{Q}^j\mathbf{B}^j + \mathbf{P}^j\mathbf{A}^j = \mathbf{1}$, Eq. (5.16), and the constraints given by Eqs. (5.14a), (5.14b) and (5.14d).

## 7.3 Example of Optimized Scaling Functions

In this section we will specify some of the free parameters in the optimization of **A** and **P**, and illustrate the resulting optimized filters.

**Define $\Lambda$** The only explicit free parameter to set is the target $\Lambda$. Since we desire the operators to act locally, we have to limit the distance of the non-zero neighbours. In other words, the $\Lambda$ matrix will be mostly zero valued, with the largest value on the diagonal, say $\gamma$. In this work the $\Lambda$ is designed so that the only non zero neighbours are the $\{v_1, v_2, f_1, f_2\}$ defined in Figure 5.6. These vertices are set to a same value $\eta = 1/2\gamma$, and so that the matrix results to be normalized.

**Independent parameters and free parameters** In the optimization of **A** and **P** all the entries of the matrices can be left completely free or they can be bound by the symmetries of the mesh, as already mentioned (a more detailed discussion can be found in Appendix B). Moreover, we can decide how many

of the independent parameters remaining from the symmetries reduction are effectively let free. By design, we would like the filter to go to zero for the points of the mesh far from the point under consideration. One option is to constrain them to be zero. With these choices we can considerably reduce the number of degrees of freedom of the problem.

**Plots of resulting scaling filters** In Figures 7.1 and 7.2 we report an example of dual scaling filters ($\mathbf{A}^1$, $\mathbf{A}^2$) obtained with this method for the spherical mesh described in Section 5.3. These figures show a comparison between the interpolating and butterfly wavelets with the optimized one. In these pictures the wavelets and scaling coming from the lifting scheme (interpolating and butterfly) are swapped for their duals (the reason will be explained in Section 8.1.2). The optimization is robust, meaning that we get the same filters by starting from very different initial conditions. A more interesting result is the Figure 7.3 which shows the combined action of cascading two filters together $\mathbf{A}^1\ \mathbf{A}^2$, that is the operation of downsampling performed during the SWF encoding, which is a representation of the dual scaling function. The resulting filters are still in number of 6, that is the number of filters at level 0, but they have a length of 66 taps.

In Figures 7.4, 7.5 and 7.6 we report the filters generated by the $\mathbf{P}$ matrices. Similar considerations to the $\mathbf{A}$ filters apply.

In general, the filters generated via the optimization tend to have less negative values and have higher values in the vicinity of the main peak, we can say that are slightly rounder, but not wider.

In Figures 7.7 and 7.8 we report the optimized filters against the swapped interpolating and butterfly ones. It is possible to notice that while the $\mathbf{A}$ and $\mathbf{P}$ filters are very similar among the different families, while the optimized $\mathbf{B}$ and $\mathbf{Q}$ filters are very different from the ones coming from the lifting scheme. Nevertheless, they all satisfy the equations (5.15) and (5.16). It is apparent that the imposed constraints are not sufficient to determine uniquely the $\mathbf{B}$ and $\mathbf{Q}$. We did not investigate further on their additional properties, since they reach their purpose, and these filters are not central in our construction.

In Chapter 8 we will analyze in more detail the effects of the differences between the filters on the SWF audio chain.

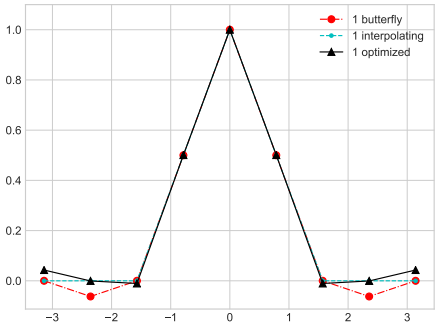



Figure 7.1: Horizontal representation of one filter of $\mathbf{A}^1$, for the butterfly, interpolating and optimized filters.

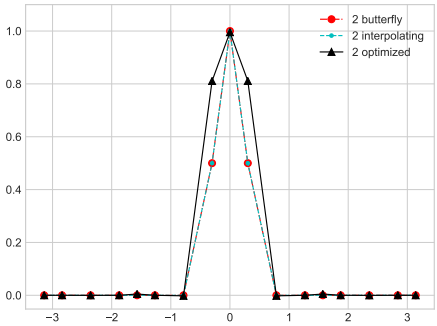



Figure 7.2: Horizontal representation of one filter of $\mathbf{A}^2$, for the butterfly, interpolating and optimized filters.

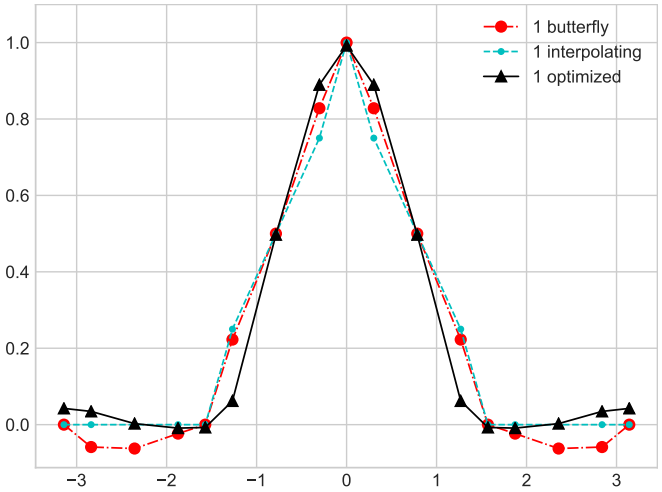



Figure 7.3: Horizontal representation of one filter of $\mathbf{A}^1\mathbf{A}^2$, for the butterfly, interpolating and optimized filters.

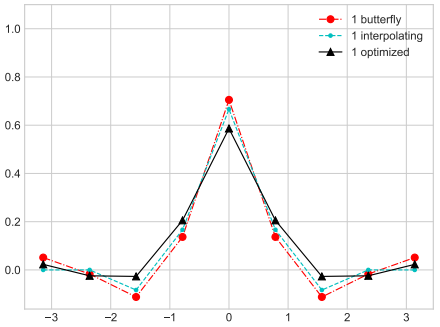



Figure 7.4: Horizontal representation of one filter of $\mathbf{P}^1$, for the butterfly, interpolating and optimized filters.

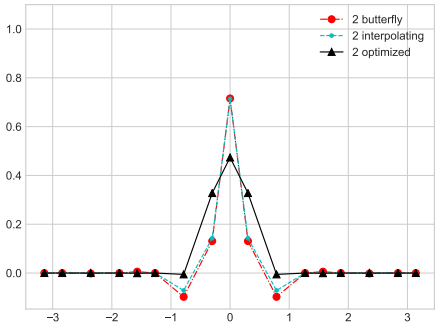



Figure 7.5: Horizontal representation of one filter of $\mathbf{P}^2$, for the butterfly, interpolating and optimized filters.

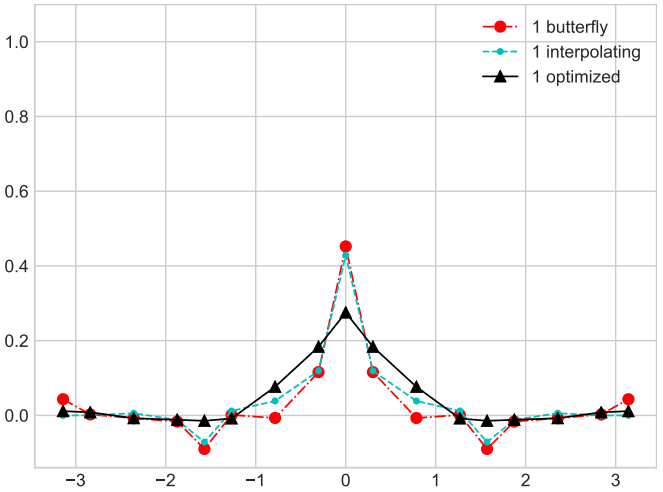



Figure 7.6: Horizontal representation of one filter of $\mathbf{P}^1\mathbf{P}^2$, for the butterfly, interpolating and optimized filters.

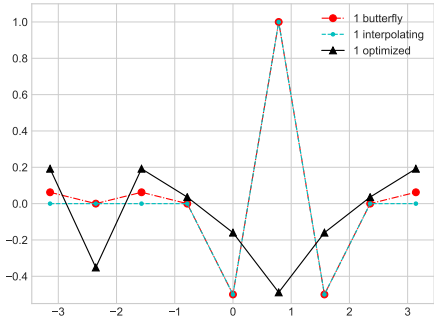



Figure 7.7: Horizontal representation of one filter of $\mathbf{Q}^1$, for the butterfly, interpolating and optimized filters.

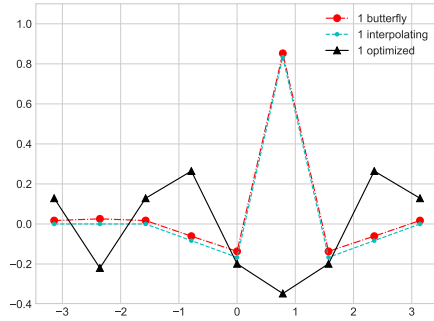



Figure 7.8: Horizontal representation of one filter of $\mathbf{B}^1$, for the butterfly, interpolating and optimized filters.

## 7.4 Summary

In this Chapter we have described how to generate numerical wavelet families in the Second Generation framework leveraging the Lifting Scheme. The definition of the lifting scheme is completely general and can be applied to any multiresolution mesh. We detailed the generic construction of the interpolating wavelet and then particularized it for a spherical mesh. Lastly, we report our method for wavelet filters optimization and we illustrate some of its outcomes.

# Part III

# Evaluation

# Chapter 8

# IMPLEMENTATION AND EVALUATION OF SPECIFIC SPHERICAL WAVELET FORMATS

In this Chapter we will describe three specific SWF implementations: VBAP-SWF based on VBAP interpolation, SINT-SFW based on the interpolating wavelet, and OPT-SWF is the result of a numerical optimized wavelet. We then proceed to compare their properties in terms of pressure, energy, velocity and intensity preservation.

## 8.1 Constructing a Specific SWF Implementation

In Section 6.3 we defined SWF as an audio encoding with three characteristics: a subdivision mesh defined over the sphere, a set of filters defining a wavelet space, a truncation level. In this Section we build three different implementations of SFW. All the three implementations have the first and third point in common: the subdivision mesh starts from an octahedron and gets refined

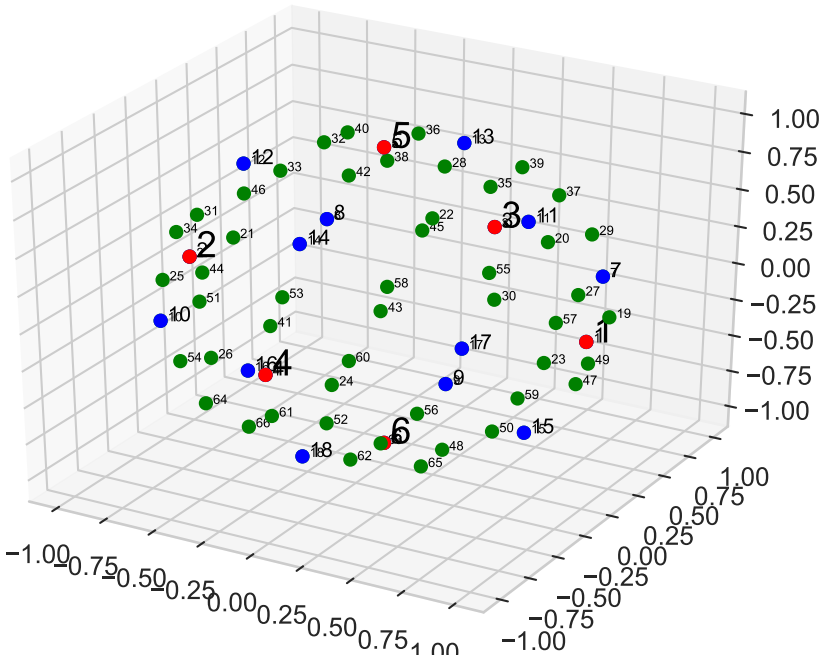



Figure 8.1: Mesh points at different levels: red level, red+blue level 1, red+blue+green level 2.

with the Loop subdivision method (see Figure 8.1). The number of channels is 6 for level 0 and 18 for level 1. The difference between the three SWF implementations is the set of filters defining the wavelet space.

To compare the three SWF implementations, we take the finest level to be 2 (66 points), and we encode a point source rotating over the horizontal plane to a 66 points mesh using VBAP to interpolate from the continuous space to the discrete mesh, $f(\theta, \phi) \rightarrow f_{66}$. The 66 channels are later downsampled to 18 or 6 using the decomposition matrices of each format. At level zero the 6 channels correspond to the six components of the coarser dual scaling function $\mathbf{c}^0$, and at level 1 the 18 channels are the 18 components of the dual scaling function at level 1, $\mathbf{c}^1$. As discussed in Section 6.3, the signals in SWF can be represented in two alternative equivalent ways: either as the scaling function at the coarser level plus a set of successively finer wavelet details,

$\{\mathbf{c}^0, \mathbf{d}^0, \dots, \mathbf{d}^{\ell-1}\}$, or as the scaling functions at the truncation level $\ell$: $\{\mathbf{c}^\ell\}$. In the first representation we need the matrices $\mathbf{A}^j$ and $\mathbf{Q}^j$, and for this Section $j = 1$. In the second representation only the downsampling $\mathbf{A}^j$ and upsampling $\mathbf{P}^j$ filters are strictly needed. By looking at $\mathbf{c}^0$ and $\mathbf{c}^1$ we investigate the properties of $\mathbf{A}^1$ and $\mathbf{A}^2$, if we limit ourselves to the downscaling operation. If we reconstruct $\mathbf{c}^1$ with the help of $\mathbf{d}^0$, $\mathbf{c}^1 = \mathbf{P}^1\mathbf{c}^0 + \mathbf{Q}^1\mathbf{d}^0$ (Eq. (6.3)), we can inspect separately the effect of $\mathbf{P}^1$ and especially $\mathbf{Q}^1$. Upsampling to $\tilde{\mathbf{c}}^0$ from $\mathbf{c}^0$, gives us the possibility to analyze the action of $\mathbf{P}^1$ alone. In Section 5.2 we presented $\mathbf{A}$ and $\mathbf{P}$ as the dual scaling filters and scaling filters respectively, and $\mathbf{c}$ as the coarse coefficients resulting from the wavelet transform. Now, we can interpret $\mathbf{c}$, $\mathbf{A}$ and $\mathbf{P}$ in a different way. If we place a set of virtual speakers on the points defining the mesh, then the signals carried by $\mathbf{c}$ would be the feeds for the speakers. If the encoded signal is a point source in a certain position, then the $\mathbf{c}$ represent the gains for each virtual speaker needed to represent a point source in that position. For this reason, and in this specific case, we could call the $\mathbf{c}$ the 'panning functions' of the SWF. With this interpretation, the matrices $\mathbf{A}$ and $\mathbf{P}$ connect layouts with different numbers of virtual speakers; in spatial audio this kind of matrices are called downmixing and upmixing matrices, respectively.

The objective of this Section is to see how different types of filters behave, in terms of pressure preservation and encoding gains for a panning around the horizontal axis. These quantities would be the ones reconstructed by a set of speakers (6 or 18) placed exactly on the mesh points location. Referring to Figure 8.2, we will compare the different flavours of SWF after the reconstruction and before the decoding to speakers.

In the following we will look at the coefficients of the encoding, or the gains of the virtual speakers, $\mathbf{c}^0$, $\mathbf{c}^1$ and $\tilde{\mathbf{c}}^0$ for different versions of SWF: in Section 8.1.1 VBAP-SWF where the interpolation between meshes is the trilinear interpolation used in VBAP; in Section 8.1.2 the SINT-SWF where the interpolation between meshes is given by the swapped-interpolating lifting wavelet, which is a modification of the interpolating wavelet family illustrated in Section 5.5; and finally in Section 8.1.3 the OPT-SWF which is the result of the optimization procedure described in Chapter 7. All the gains plots are re-

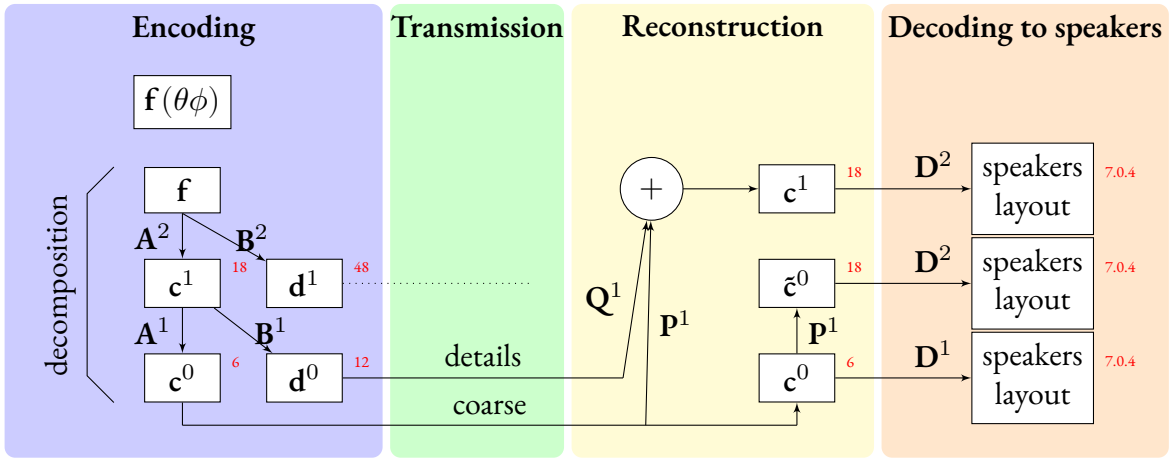



Figure 8.2: Example of a possible SWF workflow, corresponding to the particular implementation described in the text. The number of channels is noted in red.

ported in linear scale to make apparent the eventual negative gains, even if the logarithmic scale is probably more common.

### 8.1.1 VBAP-SWF

In Section 6.3 we defined the matrices $\mathbf{A}_{\text{VBAP}}$ and $\mathbf{P}_{\text{VBAP}}$ for the VBAP-SWF. These matrices are the only ones we can design with the procedure described there. To get the full set $\{\mathbf{A}_{\text{VBAP}}, \mathbf{B}_{\text{VBAP}}, \mathbf{P}_{\text{VBAP}}, \mathbf{Q}_{\text{VBAP}}, \}$, we should apply the othonormality relations (5.15) and (5.16). For several reasons, the filters **B** and **Q** typically are not obtained directly from Eqs. (5.15) and (5.16), but with procedures (e.g. Lifting Scheme) that guarantee to satisfy those equations. Since the VBAP-SWF is not obtained via the Lifting Scheme, but is somewhat built procedurally, we skip the construction of $\mathbf{B}_{\text{VBAP}}$ and $\mathbf{Q}_{\text{VBAP}} and we analyze only \mathbf{c}^0_{\text{VBAP}}$, $\mathbf{c}^1_{\text{VBAP}}$ and $\tilde{\mathbf{c}}^0_{\text{VBAP}}$ coming from $\mathbf{A}_{\text{VBAP}}$ and $\mathbf{P}_{\text{VBAP}}$.

The gains produced by the format VBAP-SWF defined in 6.3 for an horizontal panning at level 0 and 1 are reported in Figures 8.3, 8.4, 8.5. In the plots representing the gains, we represent with solid lines the (virtual) speakers located in the horizontal plane (zero elevation) and with dashed lines the (virtual) speakers with non-zero elevation.

In particular, Figure 8.4 displays the effect of $\mathbf{A}^2_{\text{VBAP}}$ alone, downsampling from 66 to 18 points. Schematically: $f(\theta, \phi) \rightarrow f_{66} \rightarrow c^1_{18}$. It is possible to notice that the downsampling is in fact a linear interpolation.

Figure 8.3 shows the joint action of $\mathbf{A}^1_{\text{VBAP}}\mathbf{A}^2_{\text{VBAP}}$ downsampling the panning from 66 to 6 points. Schematically: $f(\theta, \phi) \rightarrow f_{66} \rightarrow c^0_6$. Pressure is preserved at each application of $\mathbf{A}_{\text{VBAP}}$.

Finally, in Figure 8.5 we present the effect of upsampling the $\mathbf{c}^0$ (6 points) to $\tilde{\mathbf{c}}^0$ (18 points), which is the result of applying $\mathbf{P}^1_{\text{VBAP}}\mathbf{A}^1_{\text{VBAP}}\mathbf{A}^2_{\text{VBAP}}$ to the horizontal rotating delta, which is our signal. Schematically: $f(\theta, \phi) \rightarrow f_{66} \rightarrow c^0_6 \rightarrow \tilde{c}^0_{18}$. It is possible to note how the pressure is preserved also after this stage. The interesting fact to notice is that only the four horizontal points (or virtual speakers) at level 0 are activated, even if we effectively upsampled to level 1. This is due to the fact that the $\mathbf{P}_{\text{VBAP}}$ matrices are trivial (an identity

for the points at the lower resolution and zero elsewhere).

In this particular version of SWF all the gains are strictly positive.

### 8.1.2 SINT-SWF

We mentioned in6.3 that the VBAP-inspired wavelets are close in spirit to the interpolating wavelets, with the difference of having a different set of neighbours and with the dual and direct spaces swapped. Here we show the result of swapping the wavelets and the scaling with their duals in the case of the lifted spherical interpolating wavelets presented in Section 5.5. In the literature several types of lifted wavelets are available, other than the interpolating one, but in general they don't preserve pressure even when swapping direct and dual spaces. In this dissertation we chose the swapped interpolating wavelets for the sake of simplicity and for the analogy with the VBAP case.

In Figures 8.6, 8.7 and 8.8 we report the $\mathbf{c}^0$, $\mathbf{c}^1$, $\tilde{\mathbf{c}}^0$ for the lifted spherical interpolating wavelets (as defined in Section 5.5). It is evident that the pressure is not preserved by these $\mathbf{A}_{\text{INT}}$ and this type of scaling functions is not interesting for our application.

In Figures 8.9, 8.10 and 8.11 we show what happens to the $\mathbf{c}^0$, $\mathbf{c}^1$, $\tilde{\mathbf{c}}^0$ when we swap the dual for the direct, essentially $\mathbf{A}_{\text{SINT}} = \mathbf{P}_{\text{INT}}^{\top}$ and $\mathbf{P}_{\text{SINT}} = \mathbf{A}_{\text{INT}}^{\top}$. We will call this format SINT-SWF. Similarly to the VBAP-SWF pressure is preserved at all moments and Figure 8.10 shows the result of $\mathbf{A}_{\text{SINT}}^2$ interpolation alone. Unlike the $\mathbf{P}_{\text{VBAP}}$ the $\mathbf{P}_{\text{SINT}}$ are not trivial, and we get a meaningful upsampling, see Figure 8.11. Small negative gains are introduced in the upsampling procedure.

It is interesting to see which is the contribution of the details $\mathbf{d}^0$ when the $\mathbf{c}^1$ is obtained via the mentioned reconstruction equation $\mathbf{c}^1 = \mathbf{P}^1\mathbf{c}^0 + \mathbf{Q}^1\mathbf{d}^0$. In Figure 8.12 we show the bare wavelet coefficients $\mathbf{d}^0$. These $\mathbf{d}^0$ do not have an interpretation per-se, but they have to be uplifted via $\mathbf{Q}^1$ to make sense in the virtual speaker interpretation. In Figure 8.13 we illustrate the isolated effect of the upsampled $\mathbf{Q}^1\mathbf{d}^0$ over the 18 points of the mesh at level 1. If we look at these upsampled details in the interpretation where the mesh is a set of virtual speakers, then the gains carried by $\mathbf{Q}^1\mathbf{d}^\mathbf{0}$ are summed linearly to the

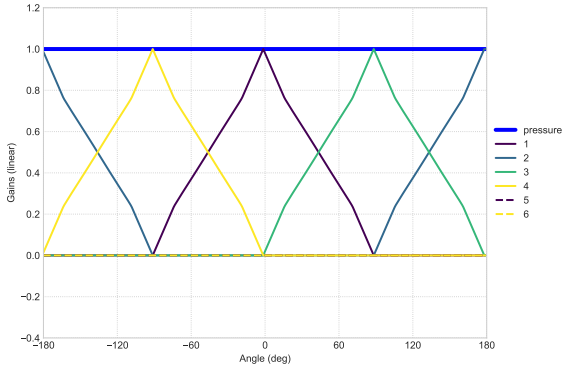



Figure 8.3: Scaling coefficients $\mathbf{c}_6^0$ for an horizontal panning of VBAP-SWF at level 0. These can be interpreted as the gains of 6 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_6^0$).

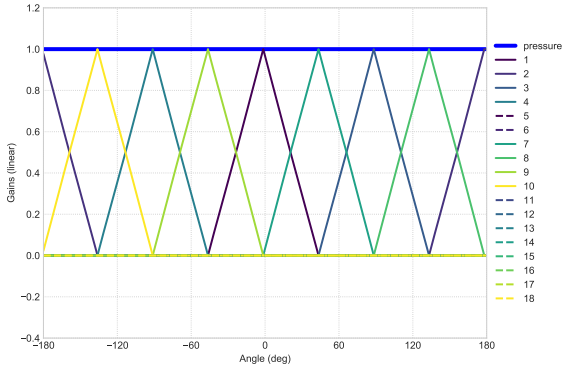



Figure 8.4: Scaling coefficients $\mathbf{c}_{18}^1$ for an horizontal panning of VBAP-SWF at level 1. These can be interpreted as the gains of 18 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_{18}^1$).

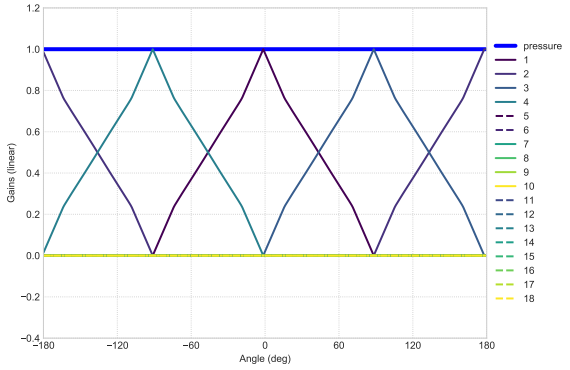



Figure 8.5: Upsampled scaling coefficients $\mathbf{c}_6^0$ for an horizontal panning of VBAP-SWF at level $\tilde{0}$. These can be interpreted as the gains of 18 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_6^0 \rightarrow \tilde{\mathbf{c}}_{18}^0$).

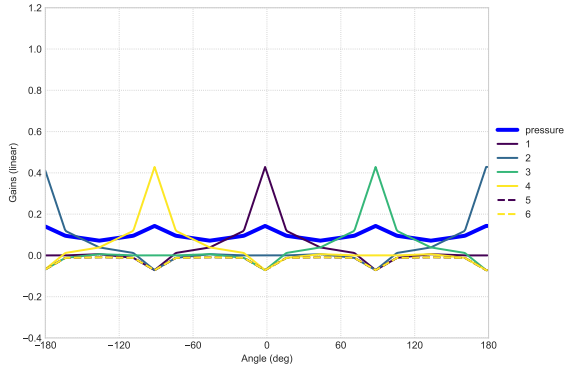



Figure 8.6: Scaling coefficients $\mathbf{c}_6^0$ for an horizontal panning of interpolating SWF at level 0. These can be interpreted as the gains of 6 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_6^0$).

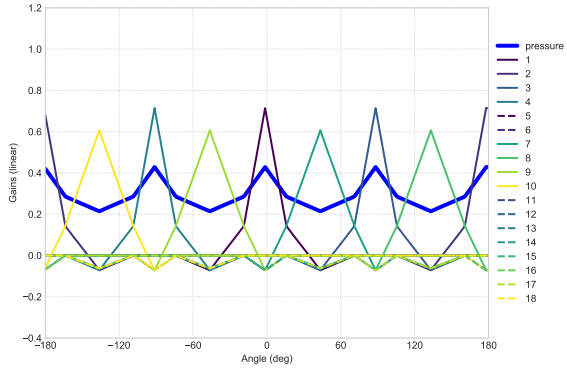



Figure 8.7: Scaling coefficients $\mathbf{c}_{18}^1$ for an horizontal panning of interpolating SWF at level 1. These can be interpreted as the gains of 18 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_{18}^1$).

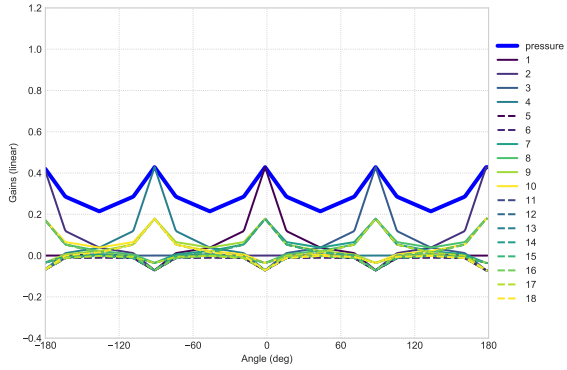



Figure 8.8: Upsampled scaling coefficients $\mathbf{c}_6^0$ for an horizontal panning of interpolating SWF at level $\tilde{0}$. These can be interpreted as the gains of 18 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_6^0 \rightarrow \tilde{\mathbf{c}}_{18}^0$).

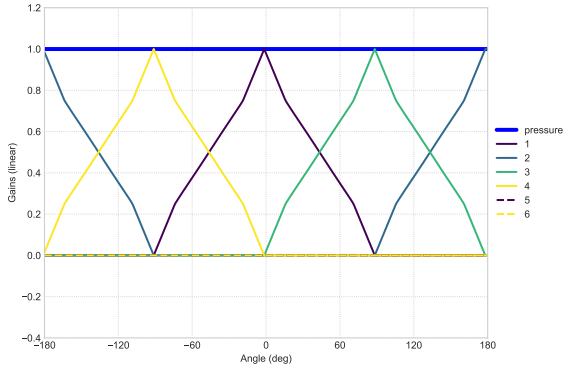



Figure 8.9: Scaling coefficients $\mathbf{c}_6^0$ for an horizontal panning of SINT-SWF at level 0. These can be interpreted as the gains of 6 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_6^0$).

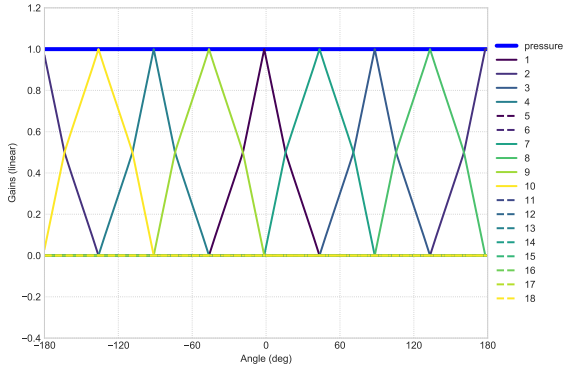



Figure 8.10: Scaling coefficients $\mathbf{c}_{18}^1$ for an horizontal panning of SINT-SWF at level 1. These can be interpreted as the gains of 18 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_{18}^1$).

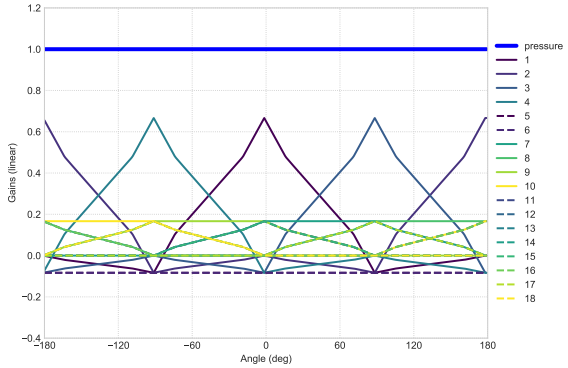



Figure 8.11: Upsampled scaling coefficients $\mathbf{c}_6^0$ for an horizontal panning of SINT-SWF at level $\tilde{0}$. These can be interpreted as the gains of 18 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_6^0 \rightarrow \tilde{\mathbf{c}}_{18}^0$).

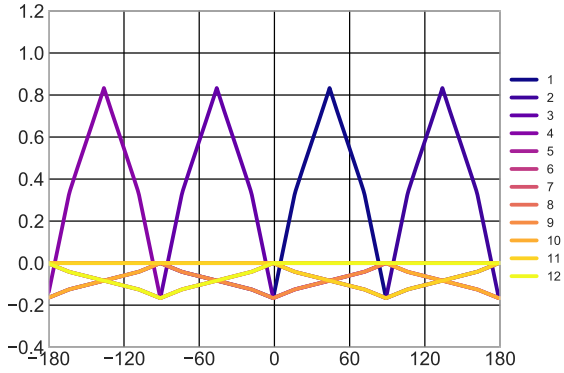



Figure 8.12: Wavelet coefficients, $\mathbf{d}^0$, for an horizontal panning of SINT-SWF at level 0.

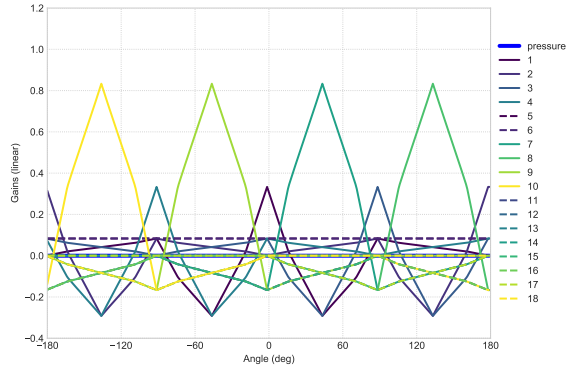



Figure 8.13: Upscaled wavelet coefficients, $\mathbf{Q}^1\mathbf{d}^0$, for an horizontal panning of SINT-SWF at level 0..

ones coming from $\mathbf{P}^1\mathbf{c}^0$. It is important to notice that these details do not carry pressure, that is already preserved by the **A** and **P** matrices.

### 8.1.3 Optimized-SWF Gains for Horizontal Panning

In Figures 8.14, 8.15 and 8.16 we report the same signals as before for the optimized set of filters of the OPT-SWF. It is possible to notice that pressure is properly preserved at all stages, similarly to what happens for VBAP-SWF and SINT-SWF. Noticeable negative gains are introduced only in the upsampling stage (Figure 8.16) and are smaller than in the equivalent processing for SINT-SWF. The panning functions have a more 'round' shape than in VBAP-SWF

and INT-SWF.

As before, it is interesting to see which is the contribution of the details $\mathbf{d}^0$ when the $\mathbf{c}^1$ is obtained via the mentioned reconstruction equation $\mathbf{c}^1 = \mathbf{P}^1\mathbf{c}^0 + \mathbf{Q}^1\mathbf{d}^0$. In Figure 8.17 we report the bare $\mathbf{d}^0$ wavelet coefficients for OPT-SWF. It is interesting to note that the $\mathbf{d}^0$ from OPT-SWF are less sparse than the one coming from SINT-SWF. This effect is due to the fact that we did not ask for special properties of $\mathbf{B}^1$ and $\mathbf{Q}^1$, a part from the othonormality with $\mathbf{A}^1$ and $\mathbf{P}^1$. Notice that the properties of $\mathbf{d}^0$ do not influence the localization properties of the wavelet-based audio format, because those are given by $\mathbf{Qd}$. The $\mathbf{d}^0$ influence only the compression qualities of the audio format. In Figure 8.18 we illustrate the isolated effect of the upsampled $\mathbf{Q}^1\mathbf{d}^0$ over the 18 points of the mesh at level 1. It is important to notice that these details do not carry pressure, that is already preserved by the $\mathbf{A}$ and $\mathbf{P}$ matrices.

The main difference with respect to the upsampled details of SINT-SWF in Figure 8.13 is the shape of the contribution of the details. In the next Section we will investigate more on this subject.

## 8.2 Reconstructed Velocity, Energy and Intensity for Different SWF Flavours

In the previous Section we have seen that the analyzed SWF flavours preserve pressure along the whole audio chain, and present some differences in panning functions' shape. In this section we will investigate the differences between the three SWF flavours, VBAP/SINT/OPT, in terms of reconstruction of some relevant observables: velocity, energy and intensity. These quantities would be the ones reconstructed by a set of speakers (6 or 18) placed exactly on the mesh points location.

We want to stress the fact that we are looking at these values as reconstructed on the points of the original mesh, as if they were virtual speakers, and no actual decoding is involved. Making a parallel with Ambisonics, it is possible to interpret the reconstructed physical quantities we are investigating as the quantities reproduced by a basic decoding with a regular layout. With

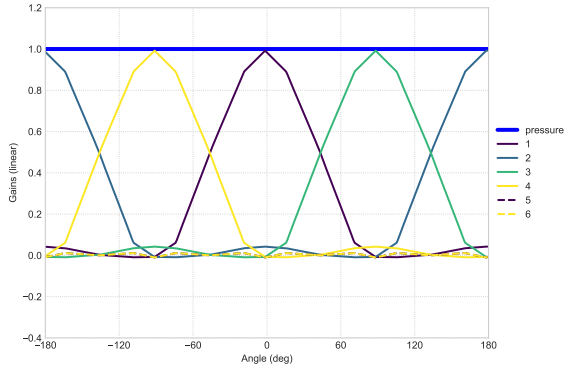



Figure 8.14: Scaling coefficients $\mathbf{c}_6^0$ for an horizontal panning of OPT-SWF at level 0. These can be interpreted as the gains of 6 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_6^0$).

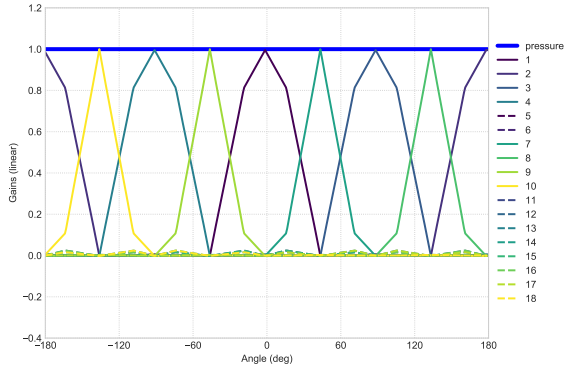



Figure 8.15: Scaling coefficients $\mathbf{c}_{18}^1$ for an horizontal panning of OPT-SWF at level 1. These can be interpreted as the gains of 18 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_{18}^1$).

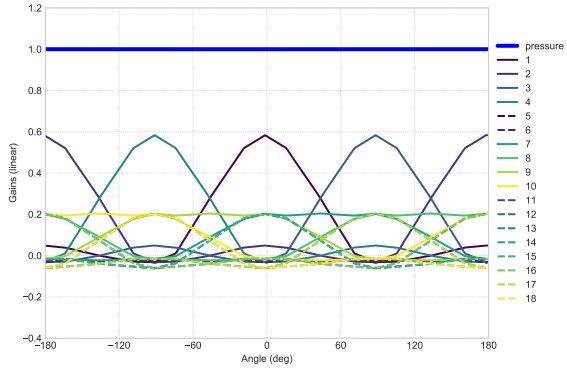



Figure 8.16: Upsampled scaling coefficients $\mathbf{c}_6^0$ for an horizontal panning of OPT-SWF at level $\tilde{0}$. These can be interpreted as the gains of 18 virtual speakers located on the mesh. ($f_{66} \rightarrow \mathbf{c}_6^0 \rightarrow \tilde{\mathbf{c}}_{18}^0$).

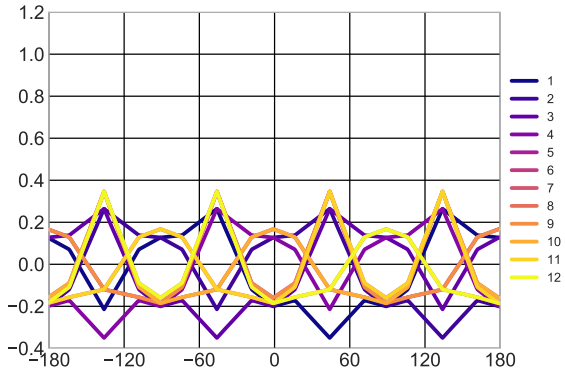



Figure 8.17: Wavelet coefficients, $\mathbf{d}^0$, for an horizontal panning of OPT-SWF at level 0.

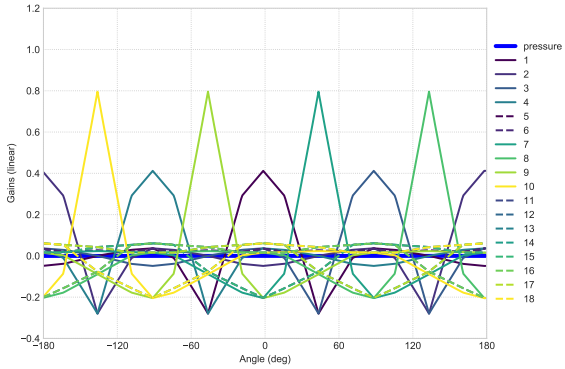



Figure 8.18: Upscaled wavelet coefficients, $\mathbf{Q}^1\mathbf{d}^0$, for an horizontal panning of OPT-SWF at level 0.

this exploration we are effectively looking at the internals of the format, inspecting which quantities are preserved and which are not along the encoding chain.

In Figures 8.19, 8.20 and 8.21 we report the velocity reconstruction, separated in its radial and transverse components, for the three SWF flavours that are object of this analysis. In Figures 8.19 and 8.20 (where only the **A** are involved) it is possible to see that the velocity (in particular the radial component) is properly reconstructed only at the points of the mesh; in between the mesh points the type of interpolation is responsible for the preserved properties. In Figure 8.21 the action of **P** adds up, modifying the reconstruction of the radial component of velocity for OPT-SWF and SINT-SWF. VBAP-SWF component remains unchanged, with respect to Figure 8.19, since $\mathbf{P}_{\text{VBAP}}$ is trivial.

Very similar considerations to the velocity reconstruction can be done for the intensity, reported in Figures 8.22, 8.23 and 8.24, with the only difference of a more relevant transverse component starting to appear in between the mesh points. The value of the radial intensity is very good already at level 0.

The downside of a good intensity reconstruction, is a quite non-uniform energy reconstruction, see Figures 8.25, 8.26 and 8.27. The variation in energy across the horizontal plane is greater or equal than 3 dB, which is expected for a pressure-preserving panning technique but not ideal for reproduction. In a typical workflow (e.g. Ambisonics) when decoding to speakers, two decoders are produced: one that aims at pressure and velocity reconstruction for low frequencies, and one that aims at energy and intensity reconstruction for high frequencies, as explained in Section 3.1. For this task we built (and later modified to cope with wavelets decoding) IDHOA. We can, for example generate a decoding that attempts to properly recover the energy, while maximizing the radial intensity (max-$r_E$). As en example, in Figures 8.28 and 8.29 we demonstrate the action of IDHOA when generating a decoder for the virtual speakers on the mesh. The result is a reduced reconstructed energy variation, less than 2 dB, at the cost of a slight reduction in the radial intensity.

These considerations are quantified numerically in Tables 8.1 and 8.2, where we report the average (and its minimum and maximum values) for each

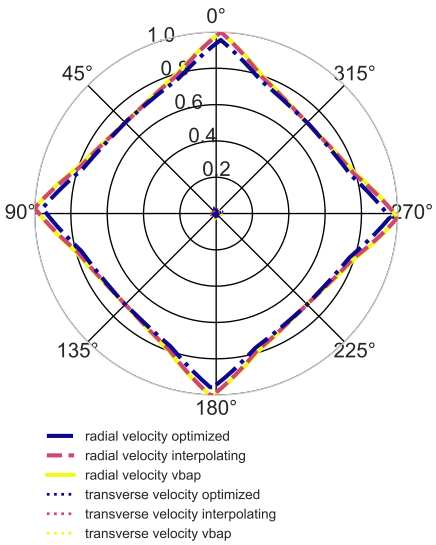



Figure 8.19: Velocity at level 0; comparison between OPT/SINT/VBAP-SWF flavours.

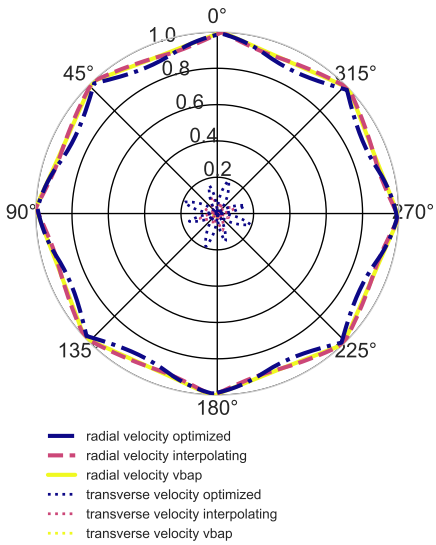



Figure 8.20: Velocity at level 1; comparison between OPT/SINT/VBAP-SWF flavours.

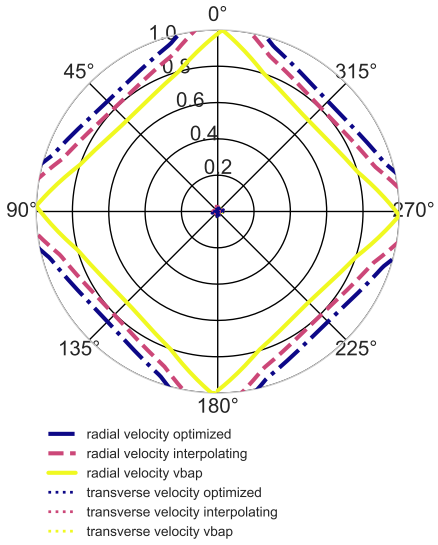



Figure 8.21: Velocity at level $\tilde{0}$; comparison between OPT/SINT/VBAP-SWF flavours.

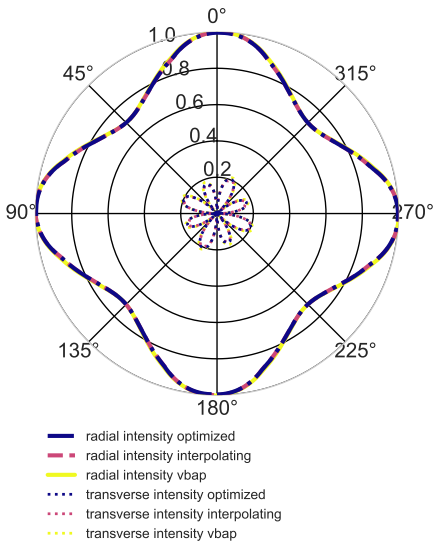



Figure 8.22: Intensity at level 0; comparison between OPT/SINT/VBAP-SWF flavours.

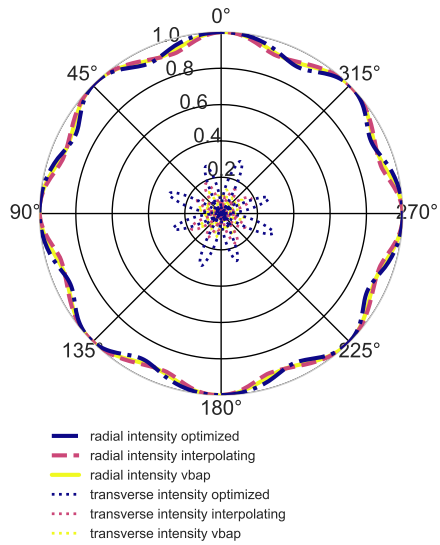



Figure 8.23: Intensity at level 1; comparison between OPT/SINT/VBAP-SWF flavours.

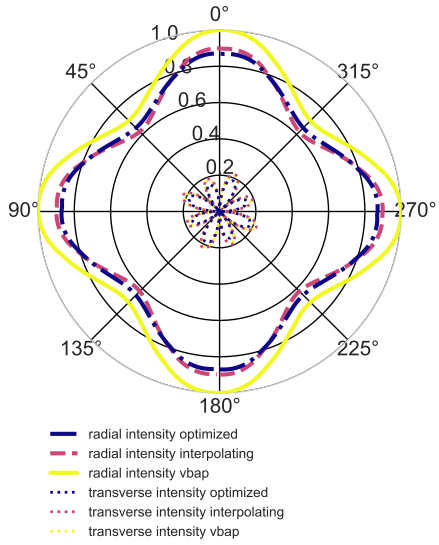



Figure 8.24: Intensity at level $\tilde{0}$; comparison between OPT/SINT/VBAP-SWF flavours.

observable and SWF version. In the last column of each table we show the action of applying the max-$r_E$ IDHOA decoding shown in Figures 8.28 for the energy and 8.29 for the intensity. It is possible to generate a different IDHOA decoder for low frequencies, that preserves pressure and optimizes the velocity, and we report the values for the improved velocity in italics in the last column of the mentioned tables. The reconstructed pressure in this decoding scheme is exactly 1, so we preferred to omit it from the Table.

As a final task, it is interesting to inspect what are the physical quantities carried by the $\mathbf{d}^0$, for SINT-SWF and OPT-SWF. In Figures 8.30 and Figures 8.31 we report the reconstructed energy and intensity carried by $\tilde{\mathbf{c}}^0 = \mathbf{P}^1\mathbf{c}^0$, $\tilde{\mathbf{d}}^0 = \mathbf{Q}^1\mathbf{d}^0$ and $\mathbf{c}^1$ for SINT-SWF. The same quantities are shown in Figures 8.32 and 8.33 for OPT-SWF. We can conclude that the physical quantities carried by the $\mathbf{d}^0$ are mainly energy and intensity, while their contribution to the pressure is zero by design.

## 8.3 Summary

In this Chapter we constructed three versions of SWF that share the same subdivision mesh, but differ in the wavelet families that constitute the SWF filters. Even if the three wavelet families are built in very different ways (VBAP interpolation, lifting scheme, numerical optimization), when looking at physical reconstructed quantities, they perform remarkably similar.

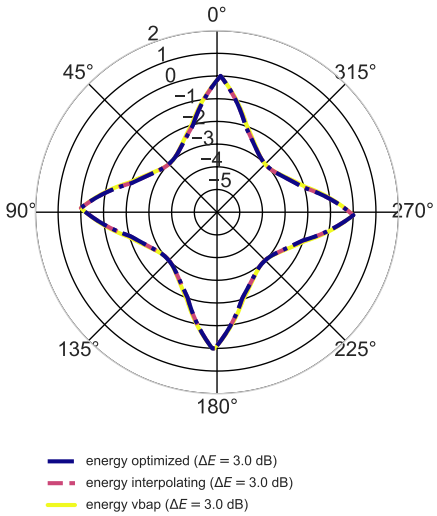



Figure 8.25: Energy at level 0; comparison between OPT/SINT/VBAP-SWF flavours.

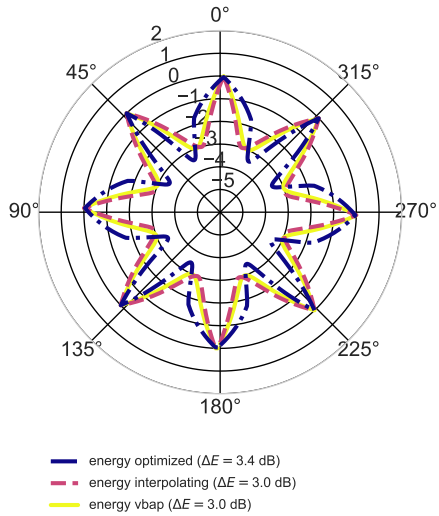



Figure 8.26: Energy at level 1; comparison between OPT/SINT/VBAP-SWF flavours.

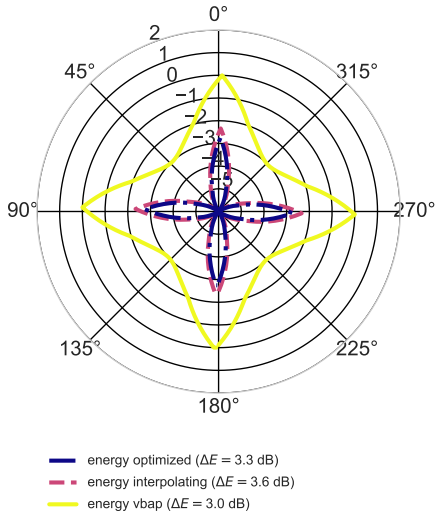



Figure 8.27: Energy at level $\tilde{0}$; comparison between OPT/SINT/VBAP-SWF flavours.

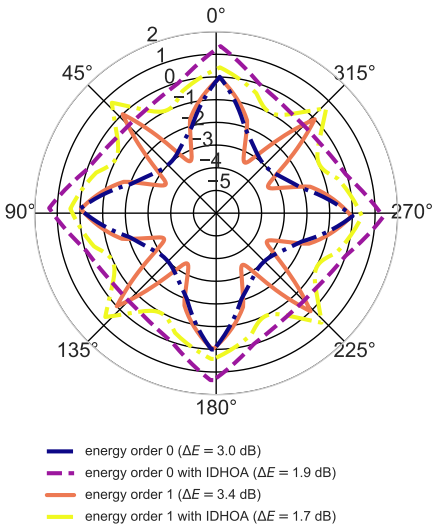



Figure 8.28: Energy as reconstructed by OPT-SWF alone and with the addition of the IDHOA decoding at levels 0 and 1.

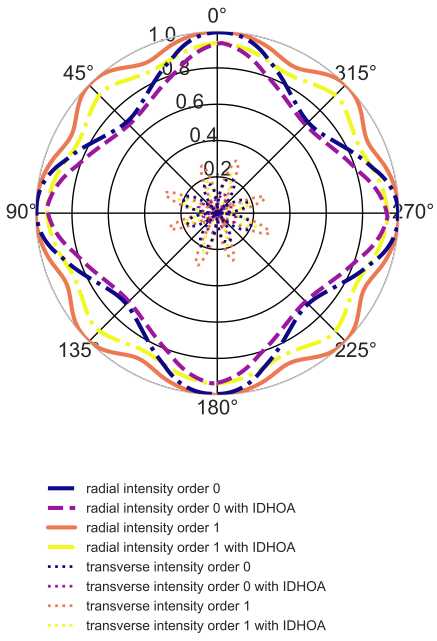



Figure 8.29: Intensity as reconstructed by OPT-SWF alone and with the addition of the IDHOA decoding at levels 0 and 1.

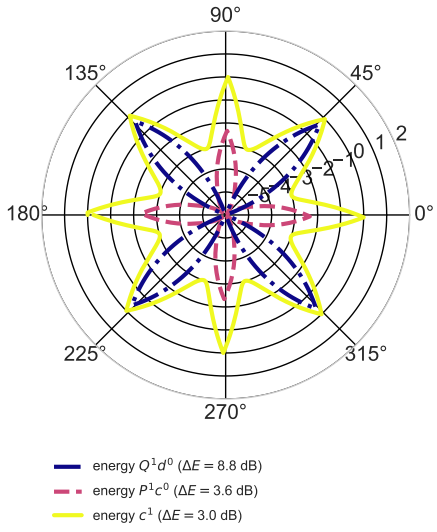



Figure 8.30: Energy contributions from $\tilde{\mathbf{c}}^0 = \mathbf{P}^1\mathbf{c}^0$ and $\tilde{\mathbf{d}}^0 = \mathbf{Q}^1\mathbf{d}^0$, which constitute $\mathbf{c}^1$, for SINT-SWF.

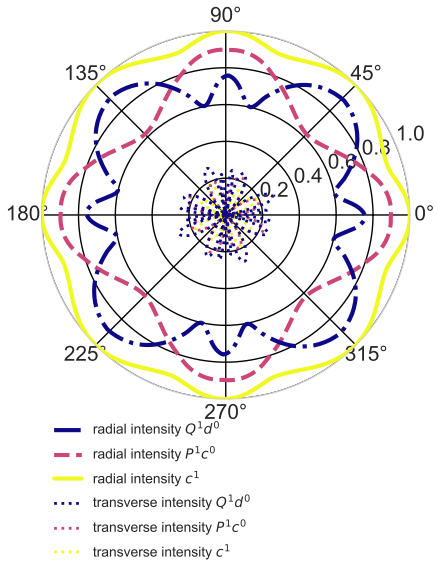



Figure 8.31: Intensity contributions from $\tilde{\mathbf{c}}^0 = \mathbf{P}^1\mathbf{c}^0$ and $\tilde{\mathbf{d}}^0 = \mathbf{Q}^1\mathbf{d}^0$, which constitute $\mathbf{c}^1$, for SINT-SWF.

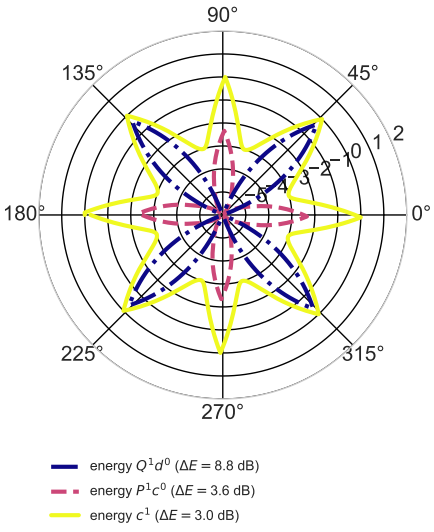



Figure 8.32: Energy contributions from $\tilde{\mathbf{c}}^0 = \mathbf{P}^1\mathbf{c}^0$ and $\tilde{\mathbf{d}}^0 = \mathbf{Q}^1\mathbf{d}^0$, which constitute $\mathbf{c}^1$, for OPT-SWF.

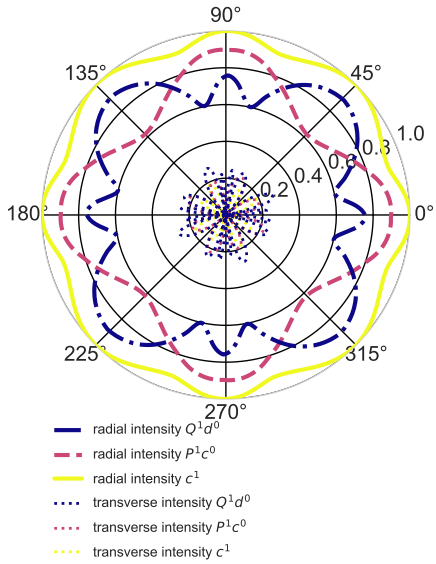



Figure 8.33: Intensity contributions from $\tilde{\mathbf{c}}^0 = \mathbf{P}^1\mathbf{c}^0$ and $\tilde{\mathbf{d}}^0 = \mathbf{Q}^1\mathbf{d}^0$, which constitute $\mathbf{c}^1$, for OPT-SWF.

| observable | VBAP-SWF | SINT-SWF | OPT-SWF | OPT-SWF + IDHOA |
|---|---|---|---|---|
| $E$ energy (dB) | $-1.91^{0.00}_{-3.01}$ | $-1.95^{0.00}_{-3.01}$ | $-1.95^{-0.01}_{-3.00}$ | $0.06^{1.36}_{-0.54}$ |
| $I_{\mathrm{R}}$ intensity | $0.85^{1.00}_{0.71}$ | $0.85^{1.00}_{0.71}$ | $0.85^{1.00}_{0.71}$ | $0.79^{0.94}_{0.69}$ |
| $I_{\mathrm{T}}$ intensity | $0.13^{0.21}_{0.00}$ | $0.12^{0.20}_{0.00}$ | $0.13^{0.20}_{0.00}$ | $0.06^{0.11}_{0.00}$ |
| $I_{\mathrm{T}}$ intensity (deg) | $7.5^{12.1}_{0.0}$ | $6.9^{11.5}_{0.0}$ | $7.5^{11.5}_{0.0}$ | $3.4^{6.3}_{0.0}$ |
| $v_{\mathrm{R}}$ velocity | $0.80^{1.00}_{0.71}$ | $0.79^{1.00}_{0.71}$ | $0.78^{0.96}_{0.71}$ | $\mathit{0.95}^{1.18}_{0.87}$ |
| $v_{\mathrm{T}}$ velocity | $0.01^{0.03}_{0.0}$ | $0.01^{0.03}_{0.0}$ | $0.01^{0.03}_{0.0}$ | $\mathit{0.02}^{0.04}_{0.00}$ |
| $v_{\mathrm{T}}$ velocity (deg) | $0.6^{1.7}_{0.0}$ | $0.6^{1.7}_{0.0}$ | $0.6^{1.7}_{0.0}$ | $\mathit{1.1}^{2.3}_{0.0}$ |

Table 8.1: Level 0 - comparison between different SWF formats. Each entry reports the average and maximum and minimum values of the specified observable, $\mathrm{avg}^{\mathrm{max}}_{\mathrm{min}}$. In italics the results for a decoder optimizing pressure and velocity.



| observable | VBAP-SWF | SINT-SWF | OPT-SWF | OPT-SWF + IDHOA |
|---|---|---|---|---|
| $E$ energy (dB) | $-1.76^{0.00}_{-3.01}$ | $-1.76^{0.00}_{-3.01}$ | $-1.67^{0.00}_{-3.44}$ | $-0.30^{0.70}_{-0.97}$ |
| $I_{\mathrm{R}}$ intensity | $0.96^{1.00}_{0.92}$ | $0.96^{1.00}_{0.92}$ | $0.96^{1.00}_{0.91}$ | $0.90^{0.94}_{0.86}$ |
| $I_{\mathrm{T}}$ intensity | $0.08^{0.14}_{0.00}$ | $0.09^{0.18}_{0.00}$ | $0.13^{0.30}_{0.00}$ | $0.09^{0.22}_{0.00}$ |
| $I_{\mathrm{T}}$ intensity (deg) | $4.6^{8.0}_{0.0}$ | $5.2^{10.4}_{0.0}$ | $7.5^{17.5}_{0.0}$ | $5.2^{12.7}_{0.00}$ |
| $v_{\mathrm{R}}$ velocity | $0.95^{1.00}_{0.92}$ | $0.95^{1.00}_{0.92}$ | $0.93^{0.99}_{0.89}$ | $\mathit{0.99}^{1.07}_{0.95}$ |
| $v_{\mathrm{T}}$ velocity | $0.02^{0.02}_{0.00}$ | $0.04^{0.10}_{0.00}$ | $0.09^{0.19}_{0.0}$ | $\mathit{0.09}^{0.20}_{0.00}$ |
| $v_{\mathrm{T}}$ velocity (deg) | $1.1^{1.1}_{0.00}$ | $2.3^{5.7}_{0.00}$ | $5.2^{10.9}_{0.0}$ | $\mathit{5.2}^{11.5}_{0.0}$ |

Table 8.2: Level 1 - comparison between different SWF formats. Each entry reports the average and maximum and minimum values of the specified observable, $\mathrm{avg}^{\mathrm{max}}_{\mathrm{min}}$. In italics the results for a decoder optimizing pressure and velocity.

# Chapter 9

# OBJECTIVE EVALUATION OF THE DECODING OF SWF FLAVOURS AND AMBISONICS

In this Chapter we compare a VBAP-SWF and OPT-SWF implementations with Higher Order Ambisonics, both decoded with IDHOA to a standard layout.

## 9.1 Objective Evaluation of VBAP-SWF, OPT-SWF and Ambisonics for the 7.0.4 Layout

As a first comparison, the chosen destination speakers' layout is a standard 7.0.4, meaning: 7 speakers on the horizontal plane, 0 LFE, 4 ceiling speakers, located as shown in Figure 9.1. The benefits of this layout are that it an industry-standard layout with loudspeakers above the horizontal plane, that is meaningful and can be easily replicated. With meaningful we mean than has a sufficient number of elevated speakers to distinguish front, back, left and right in the upper part of the layout (which is not the case for the 5.0.2). Moreover, the number of loudspeakers is still moderate with respect to other industry standards, e.g. 9.0.6, 22.2, and it is increasingly common in mixing facilities.

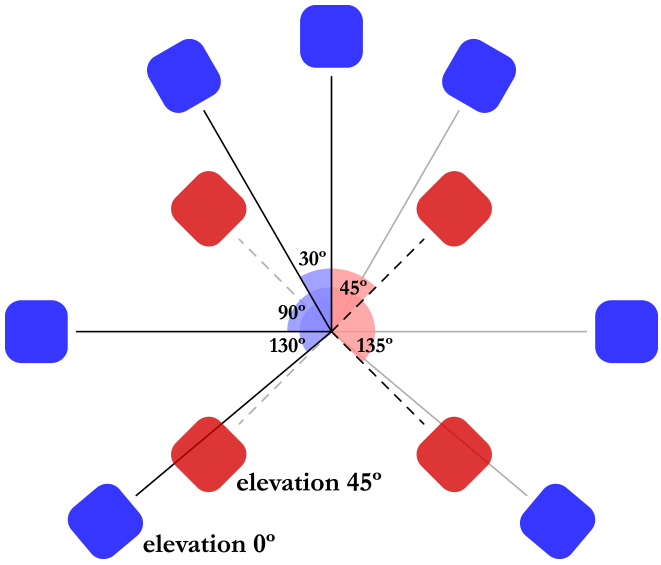



Figure 9.1: Detailed view of 7.0.4 speakers' layout.

With reference to the signal processing scenarios described in the introduction of this Chapter, for the sake of clarity we can represent them schematically as follows. The first scenario $f \rightarrow f_{66} \rightarrow c_6 \rightarrow s_{7.0.4}$:

$$
\begin{aligned}
f(\theta, \phi) &\rightarrow \mathbf{f}_{66} \\
\mathbf{c}_6^0 &= \mathbf{A}_{6\times 18}^1 \mathbf{A}_{18\times 66}^2 \mathbf{f}_{66}^2 \\
\mathbf{s}_{7.0.4}^0 &= \mathbf{D}_{7.0.4\times 6}^0 \mathbf{c}_6^0
\end{aligned}
$$

The second scenario $f \rightarrow f_{66} \rightarrow c_{18} \rightarrow s_{7.0.4}$:

$$
\begin{aligned}
f(\theta, \phi) &\rightarrow \mathbf{f}_{66} \\
\mathbf{c}_{18}^1 &= \mathbf{A}_{18\times 66}^2 \mathbf{f}_{66}^2, \text{ or} \\
\mathbf{s}_{7.0.4}^1 &= \mathbf{D}_{7.0.4\times 18}^1 \mathbf{c}_{18}^1
\end{aligned}
$$

| type | order | # channels |
|---|---|---|
| Ambisonics | 1 | 4 |
| Ambisonics | 2 | 9 |
| Ambisonics | 3 | 16 |

| type | level | # channels |
|---|---|---|
| Wavelet | 0 | 6 |
| Wavelet | 1 | 18 |
| Wavelet | $\tilde{0}$ | $6\,(\tilde{18})$ |

Table 9.1: Comparison of number of channels per order/level for Ambisonics, on the left, and Wavelets, on the right.

Finally, the third scenario $f \rightarrow f_{66} \rightarrow c_6 \rightarrow \tilde{c}_{18} \rightarrow s_{7.0.4}$:

$$
\begin{aligned}
f\left(\theta, \phi\right) &\rightarrow \mathbf{f}_{66} \\
\mathbf{c}_6^0 &= \mathbf{A}_{6\times 18}^1 \mathbf{A}_{18\times 66}^2 \mathbf{f}_{66}^2 \\
\tilde{\mathbf{c}}_{18}^0 &= \mathbf{P}_{18\times 6}^1 \mathbf{c}_6^0 \\
\tilde{g}_{7.0.4}^0 &= \mathbf{D}_{7.0.4\times 18}^1 \tilde{\mathbf{c}}_{18}^0
\end{aligned}
$$

The stages outlined here, encoding (consisting of interpolation and decomposition), transmission, upsampling (optional) and decoding to speakers, are graphically illustrated in Figure 8.2 with the number of channels annotated in red close to each box. Note that the downsampling, upsampling and decoding steps are kept separate for clarity, but they can be performed as one single matrix product.

Given the difference in number of channels between Ambisonics orders and SWF levels, see Table 9.1 it is difficult to fairly compare the two. Moreover, the destination layout has 11 speakers and theoretically, for a regular Ambisonics layout— only up to second order 3D Ambisonics could be decoded to it. Given these facts, we show the activation gains, reconstructed energy and intensity in the horizontal plane only (for the sake of brevity) for the following formats decoded to 7.0.4: $1^{st}$, $2^{nd}$, $3^{rd}$ order Ambisonics, levels 0, 1, $\tilde{0}$ in OPT-SWF and VBAP-SWF. For the SWF (both flavours) we report the resulting gains obtained with two fairly extreme configurations of IDHOA, and hence two very different decoders. The first decoder is designed to mimic the smoothness of Ambisonics (at $1^{st}$, $2^{nd}$ order), we call it *smooth* in the following. The second one is built with the intent of activating less (neighbouring)

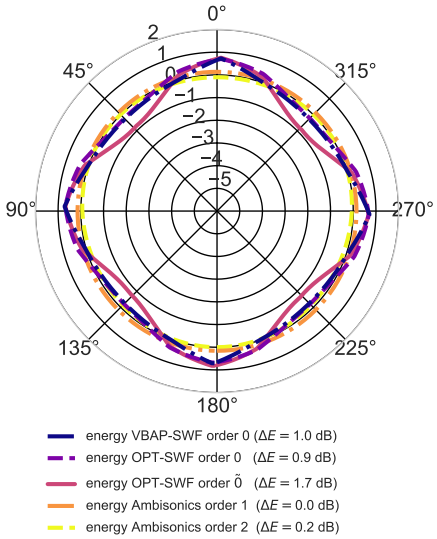



Figure 9.2: Energy comparison for levels and orders with similar channel count: SWF at level 0 with the *smooth* decoding preset and Ambisonics at order 1 and 2, decoded to a 7.0.4 layout.

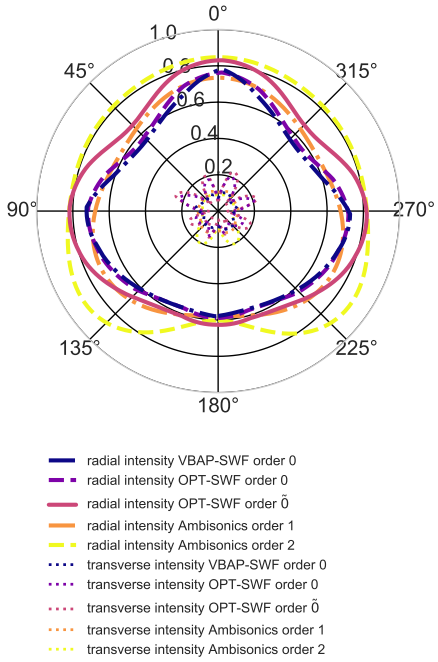



Figure 9.3: Intensity comparison for levels (with the *smooth* decoding preset) and orders with similar channel count. The point and dash lines represent radial intensity and the dashed ones the transverse intensity component.

speakers possible, mimicking a VBAP-like behaviour, and we call it *focus*. The decoders are obtained by balancing the terms of energy reconstruction, and the request for focused sources represented by the radial intensity, both described in Section 3.1.

### 9.1.1 Comparison between SWF Levels 0 and $\tilde{0}$ and Ambisonics Orders 1 and 2

For VBAP-SWF the differences between level 0 and level $\tilde{0}$ are given only by the decoding matrix **D**, and not by the upsampling, since the matrix **P** generated

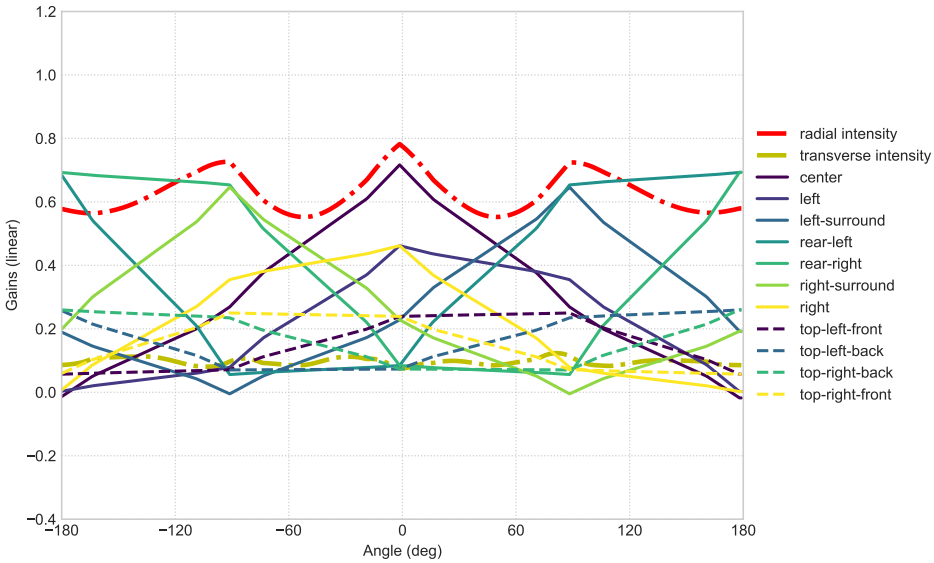



Figure 9.4: Horizontal panning for VBAP-SWF at level 0 decoded to 7.0.4 layout, using the *smooth* decoding preset.

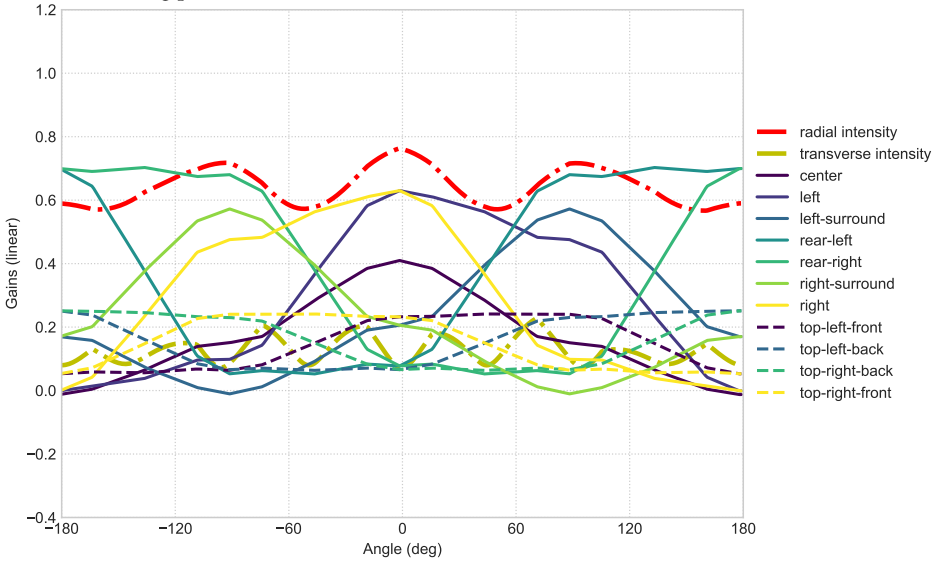



Figure 9.5: Horizontal panning for OPT-SWF at level 0 decoded to 7.0.4 layout, using the *smooth* decoding preset.

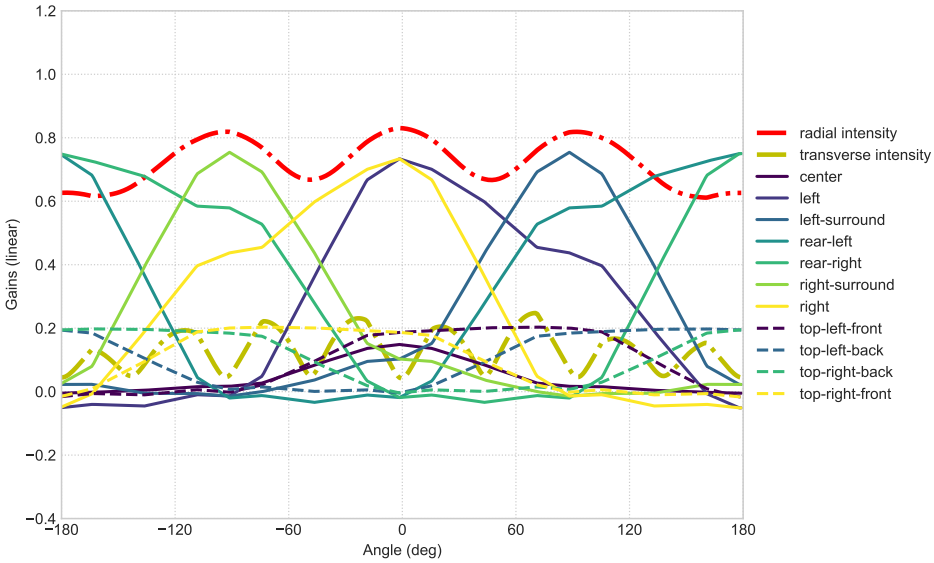



Figure 9.6: Horizontal panning for OPT-SWF at level $\tilde{0}$ decoded to 7.0.4 layout, using the *smooth* decoding preset.

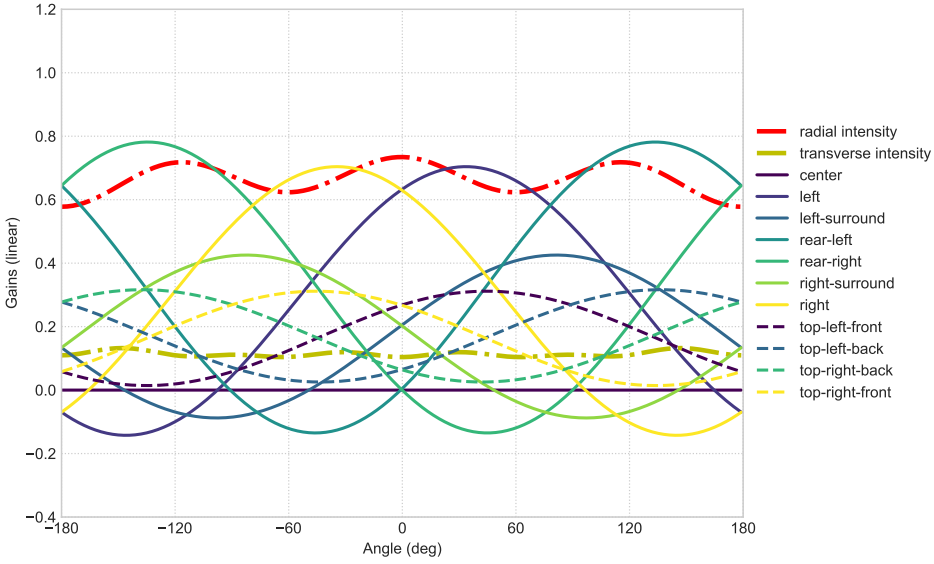



Figure 9.7: Horizontal panning for Ambisonics at order 1 decoded to 7.0.4 layout.

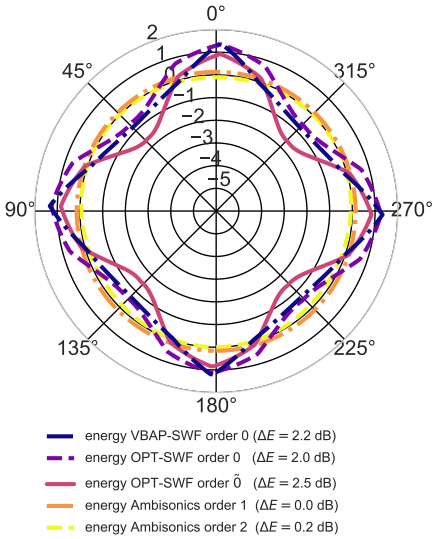



Figure 9.8: Energy comparison for levels and orders with similar channel count: SWF at level 0 with the *focus* decoding preset and Ambisonics at order 1 and 2, decoded to a 7.0.4 layout.

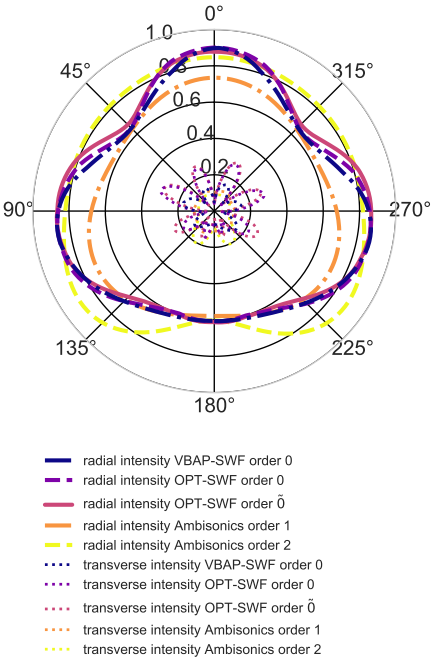



Figure 9.9: Intensity comparison for levels (with the *focus* decoding preset) and orders with similar channel count. The dashed lines represent radial intensity and the dotted ones the transverse intensity component.

by VBAP is trivial. For this reason (and for a clearer presentation), for VBAP-SWF we show only level 0, e.g. Figure 9.4.

For OPT-SWF, instead, the differences are given by the combination of **P** and **D** matrices. Figures 9.5 and 9.6 show the activation gains for the *smooth* decoding preset, and Figures 9.11 and 9.12 show the activation gains for the *focus* decoding preset.

It is interesting to observe how the two SWF formats compare with Ambisonics in terms of apparent source width, which is related to the radial intensity reported in Figures 9.3 and 9.9 for the two decodings. Figure 9.3, for the *smooth* decoder, shows that VBAP-SWF, OPT-SWF-0 and Ambisonics-1 are

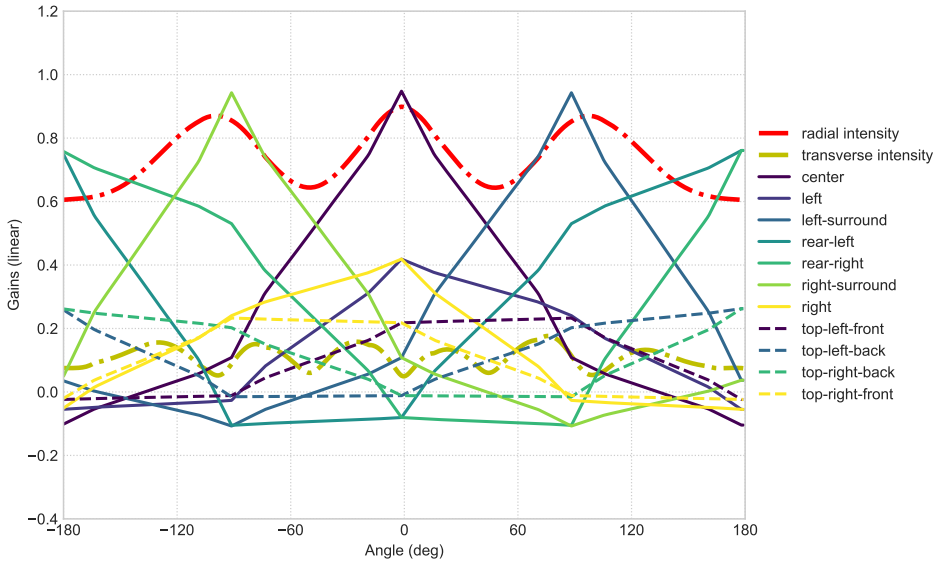



Figure 9.10: Horizontal panning for VBAP-SWF at level 0 decoded to 7.0.4 layout, using the *focus* decoding preset.

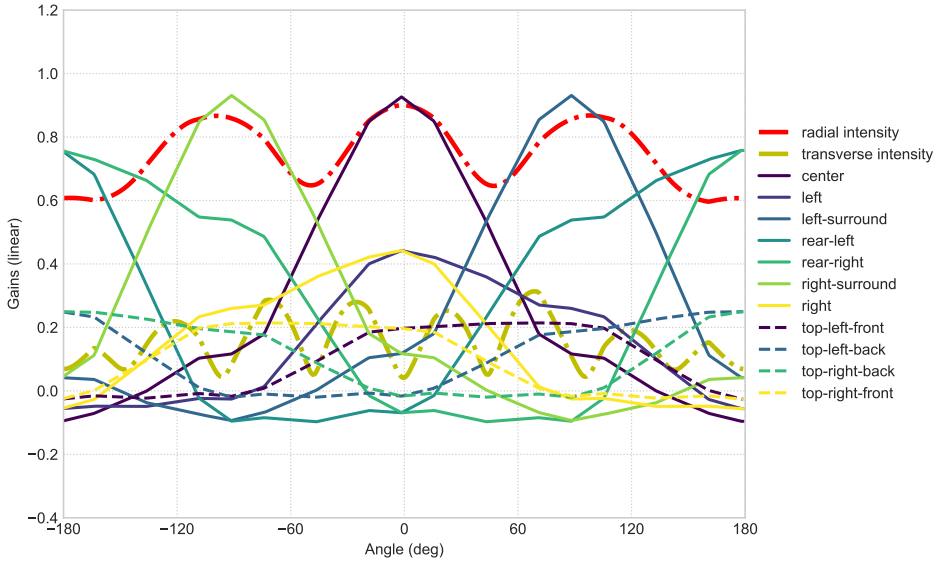



Figure 9.11: Horizontal panning for OPT-SWF at level 0 decoded to 7.0.4 layout, using the *focus* decoding preset.

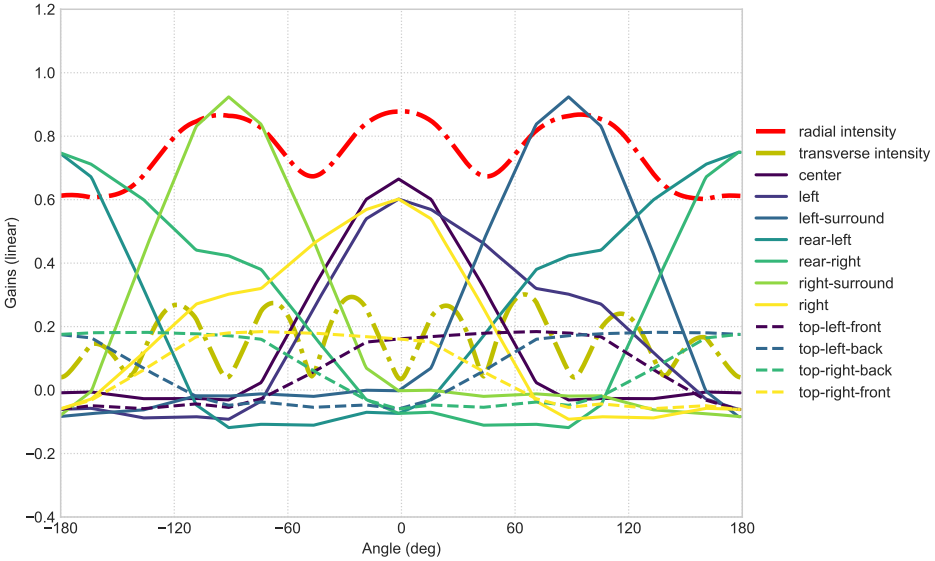



Figure 9.12: Horizontal panning for OPT-SWF at level $\tilde{0}$ decoded to 7.0.4 layout, using the *focus* decoding preset.

very similar. Interestingly, OPT-SWF-$\tilde{0}$ performs better than the plain level 0 and performs in some areas (front and sides) as Ambisonics-2. The improvement comes with a worsened reconstruction error, see Figure 9.2. Figure 9.9, for the *focus* decoder, shows that it is possible to improve SWF's focusing of the sources in some areas (front and sides, where the points of the original mesh are located) at the expense of some worsening in the energy reconstruction. Given the activation gains for this decoder, Figures 9.10, 9.11 and 9.12, and the experience during listening, this decoder sounds more "jumpy" than the *smooth* one. This effect is given by the fact that the L and R speakers are less active during the transition to the side-surrounds, with respect to the *smooth* decoder.

All these considerations are summarized in Tables 9.2 and 9.3. These observations are confirmed in informal listening tests described in Section 9.1.3.

Note that it is legit to compare level $\tilde{0}$ with Ambisonics order 1 and 2 because level $\tilde{0}$ has effectively only 6 channels, since the upsampling operation

consists of a matrix product that can be precalculated and embedded into the decoding matrix, leading to a new decoding matrix.

Ambisonics, at all orders considered here, is the one with largest negative gains. VBAP-SWF, by construction, has no negative gains, but the decoding to speakers can introduce them. OPT-SWF has some negative gains embedded in the downsampling and upsampling matrix, and the decoding to speakers can further increase them. For these reasons, in general, the two SWF flavours are better behaved than Ambisonics (order 1 and 2) when listening out of the sweet spot and are closer to a pure amplitude panner.

### 9.1.2 Comparison between SWF Level 1 and Ambisonics Orders 2 and 3

In this case VBAP-OPT-1 (Figures Figures 9.14, 9.15) and OPT-SWF-1 (Figures 9.14, 9.16) performs comparably to Ambisonics-2 (Figures 9.14, 9.17) for the *smooth* decoding. For the *focus* decoding SWF-1 performs on par (or slightly better) than Ambisonics-3, see Figures 9.20, 9.22 and 9.18. In particular, looking at the linear gains plots, it is possible to notice that SWF-1 *focus* makes use of the central loudspeaker in a neater way than Ambisonics-3 does.

All these considerations are summarized in Tables 9.4 and 9.5.

In general, the price to pay for a greater radial intensity is a worsened reconstructed energy. This effect is highlighted in the polar plots showing the reconstructed energy by reporting the $E_{max} - E_{min} = \Delta E$ in the legend. At the levels indicated here, $< 2$ dB, this effect does not seem to be noticeable, but further investigation is needed.

### 9.1.3 Informal Listening

During the evaluation of the formats and decodings, we carried out some informal listening tests. The tests where performed in a critical listening room with RT60 below 0.25 s above 200 Hz and 0.4 s below. The listening tests assessed the quality of the decoders and the different audio chains.

The differences in the perceived localization properties of the audio chains described in Sections 9.1.1 and 9.1.2 are confirmed also during the subjective

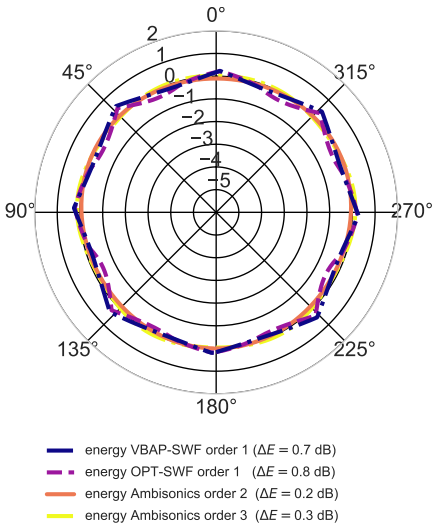



Figure 9.13: Energy comparison for levels and orders with similar channel count: SWF at level 1 with the *smooth* decoding preset and Ambisonics at order 2 and 3, decoded to a 7.0.4 layout.

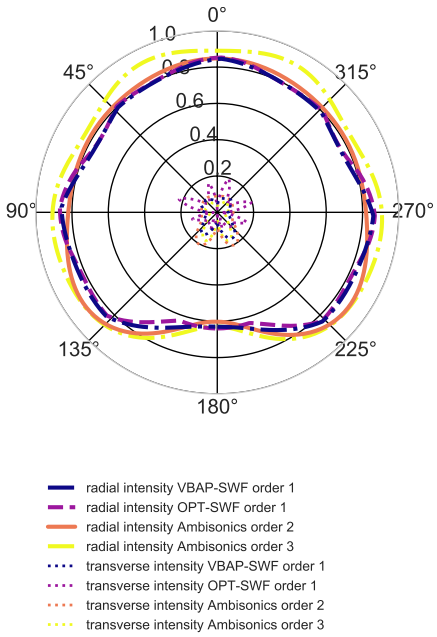



Figure 9.14: Intensity comparison for levels (with the *smooth* decoding preset) and orders with similar channel count. The point and dash lines represent radial intensity and the dashed ones the transverse intensity component.

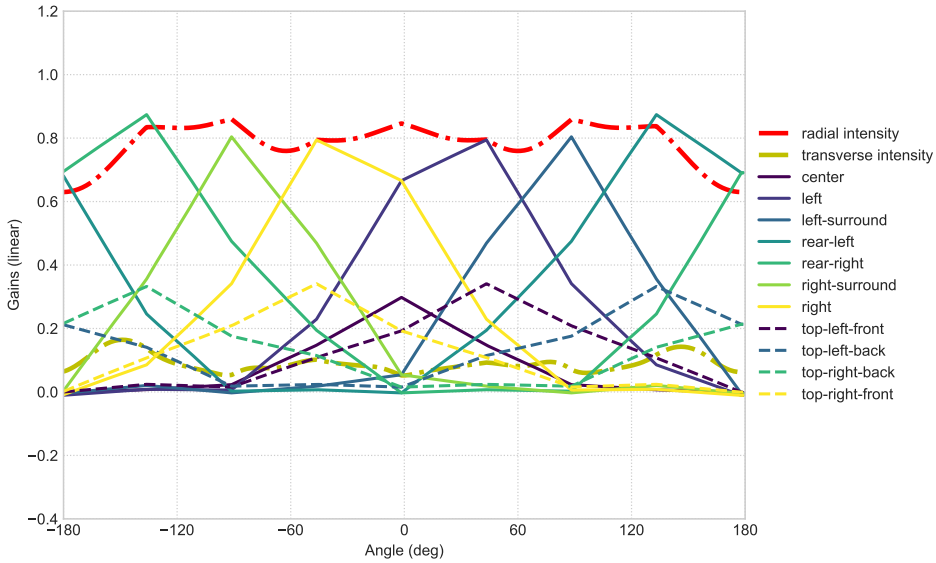



Figure 9.15: Horizontal panning for VBAP-SWF at level 1 decoded to 7.0.4 layout, using the *smooth* decoding preset.

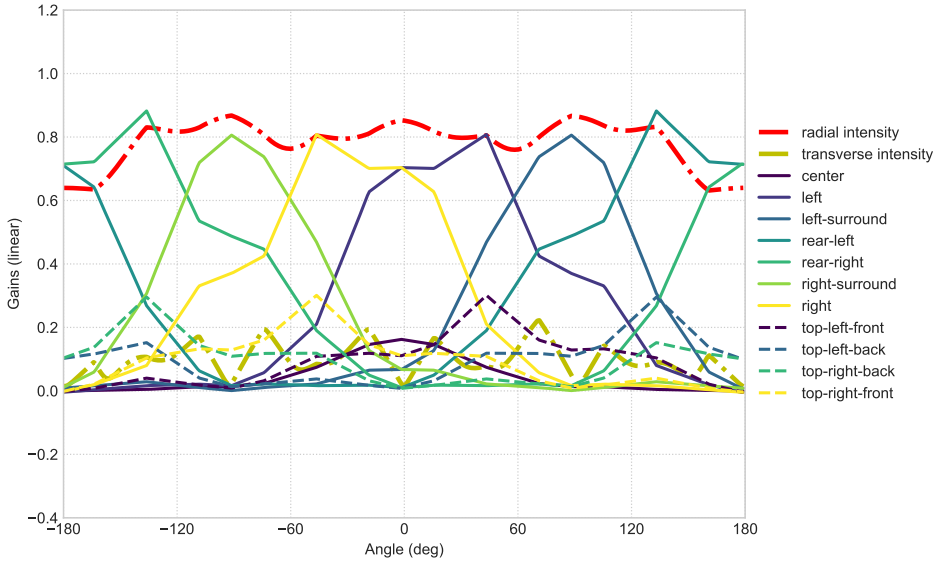



Figure 9.16: Horizontal panning for OPT-SWF at level 1 decoded to 7.0.4 layout, using the *smooth* decoding preset.

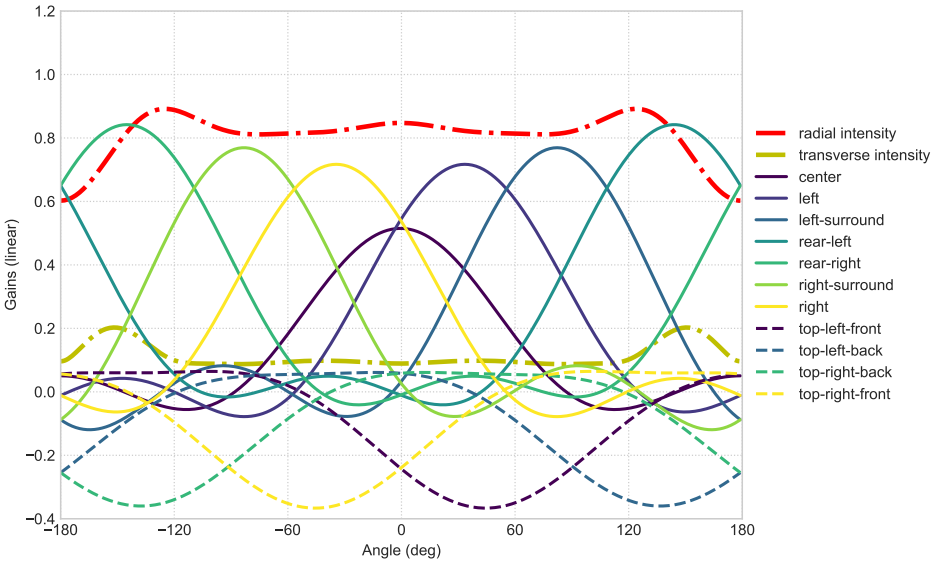



Figure 9.17: Horizontal panning for Ambisonics at order 2 decoded to 7.0.4 layout.

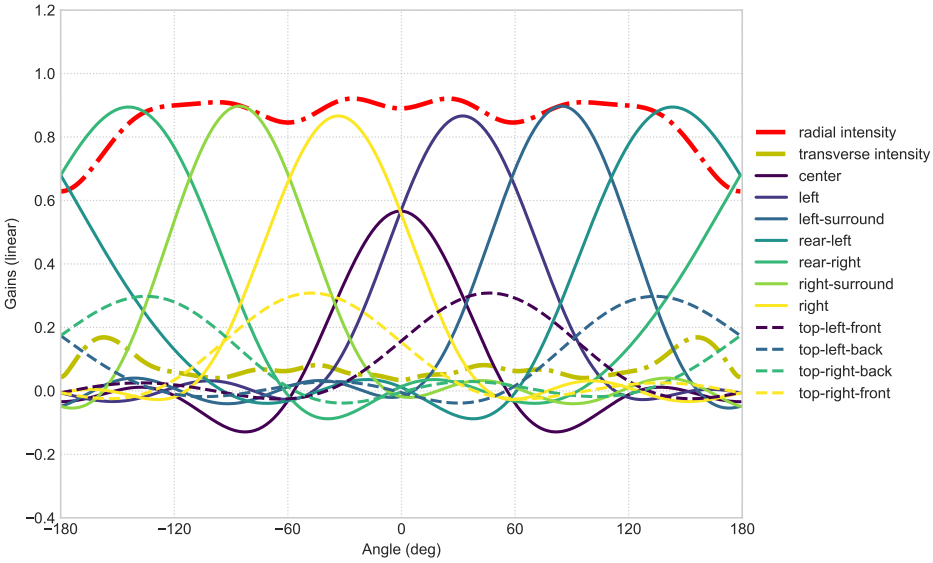



Figure 9.18: Horizontal panning for Ambisonics at order 3 decoded to 7.0.4 layout.

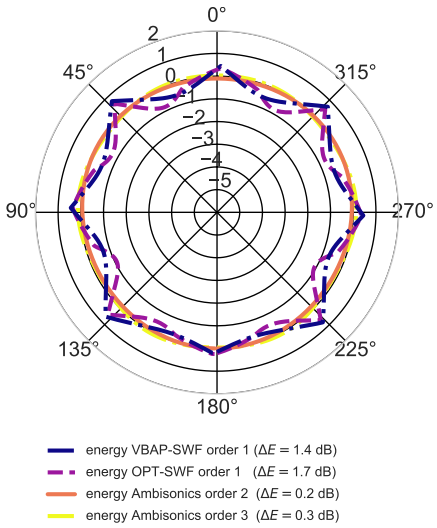



Figure 9.19: Energy comparison for levels and orders with similar channel count: SWF at level 1 with the *focus* decoding preset and Ambisonics at order 2 and 3, decoded to a 7.0.4 layout.

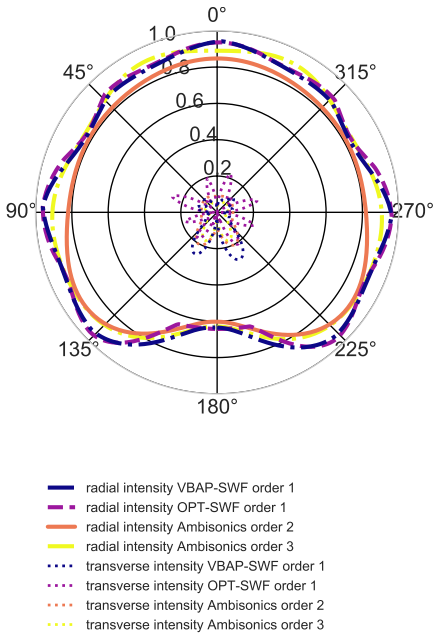



Figure 9.20: Intensity comparison for levels (with the *focus* decoding preset) and orders with similar channel count. The dashed lines represent radial intensity and the dotted ones the transverse intensity component.

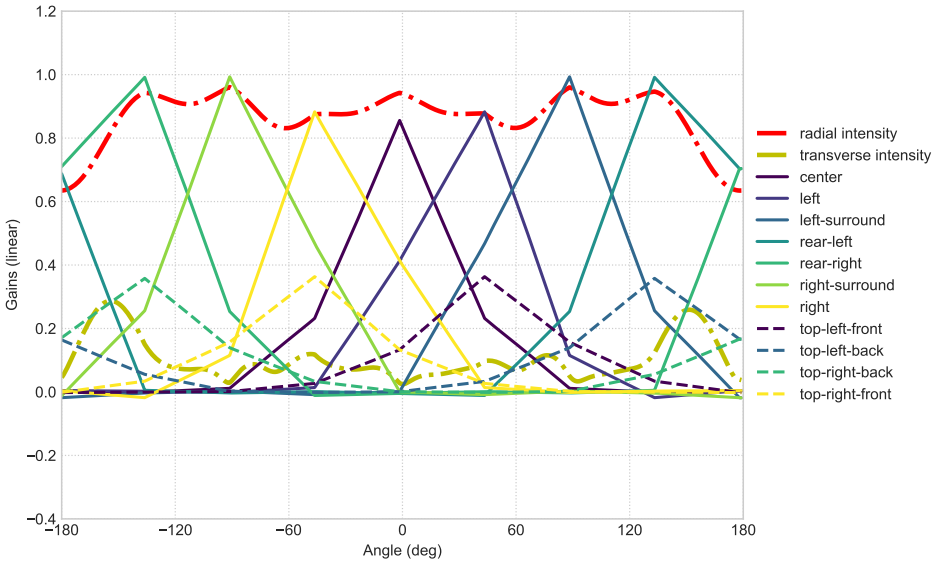



Figure 9.21: Horizontal panning for VBAP-SWF at level 1 decoded to 7.0.4 layout, using the *focus* decoding preset.

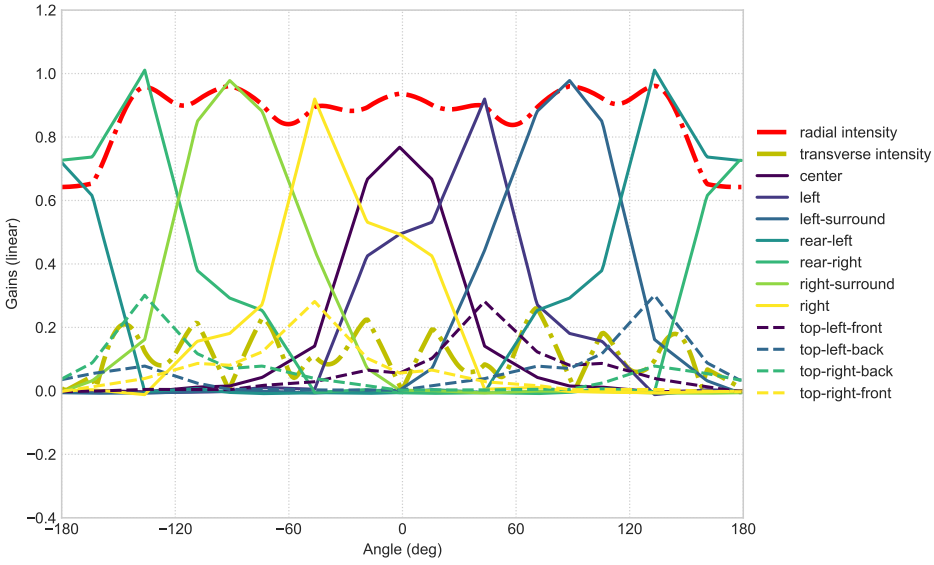



Figure 9.22: Horizontal panning for OPT-SWF at level 1 decoded to 7.0.4 layout, using the *focus* decoding preset.

| observable | VBAP-SWF 0 | OPT-SWF 0 | OPT-SWF $\tilde{0}$ | Ambi 1 | Ambi 2 |
|---|---|---|---|---|---|
| $E$ energy (dB) | $0.05^{0.73}_{-0.27}$ | $0.19^{0.70}_{-0.22}$ | $-0.07^{0.82}_{-0.84}$ | $0.14^{0.16}_{0.14}$ | $-0.06^{0.03}_{-0.16}$ |
| $I_{\mathrm{R}}$ intensity | $\mathit{0.63}^{0.78}_{0.55}$ | $\mathit{0.64}^{0.76}_{0.57}$ | $0.72^{0.83}_{0.61}$ | $\mathit{0.66}^{0.73}_{0.58}$ | $\mathbf{0.81}^{0.89}_{0.60}$ |
| $I_{\mathrm{T}}$ intensity | $0.10^{0.12}_{0.08}$ | $0.13^{0.23}_{0.07}$ | $0.13^{0.25}_{0.04}$ | $0.11^{0.13}_{0.10}$ | $0.11^{0.20}_{0.09}$ |
| $I_{\mathrm{T}}$ intensity (deg) | $5.7^{6.9}_{4.6}$ | $7.5^{13.3}_{4.0}$ | $7.47^{14.5}_{2.3}$ | $6.3^{7.5}_{5.5}$ | $6.3^{11.5}_{5.2}$ |

Table 9.2: Summary table for the comparison between SWF levels 0 and $\tilde{0}$ and Ambisonics order 1 and 2, decoded to a 7.0.4 layout with the smooth preset. Each entry reports the average and maximum and minimum values of the specified observable, $\mathrm{avg}^{\mathrm{max}}_{\mathrm{min}}$. Highlighted in italic the values of mean radial intensity that are similar across different formats. Highlighted in blod the highest value for the mean radial intensity.



| observable | VBAP-SWF 0 | OPT-SWF 0 | OPT-SWF $\tilde{0}$ | Ambi 1 | Ambi 2 |
|---|---|---|---|---|---|
| $E$ energy (dB) | $-0.04^{1.40}_{-0.80}$ | $0.30^{1.35}_{-0.63}$ | $-0.33^{0.92}_{-1.62}$ | $0.14^{0.16}_{0.14}$ | $-0.06^{0.03}_{-0.16}$ |
| $I_{\mathrm{R}}$ intensity | $\mathit{0.74}^{0.90}_{0.60}$ | $\mathit{0.75}^{0.90}_{0.60}$ | $\mathit{0.75}^{0.88}_{0.60}$ | $0.67^{0.73}_{0.58}$ | $\mathbf{0.81}^{0.89}_{0.60}$ |
| $I_{\mathrm{T}}$ intensity | $0.11^{0.18}_{0.05}$ | $0.16^{0.31}_{0.04}$ | $0.16^{0.30}_{0.03}$ | $0.11^{0.13}_{0.10}$ | $0.11^{0.20}_{0.09}$ |
| $I_{\mathrm{T}}$ intensity (deg) | $6.3^{10.4}_{2.9}$ | $9.2^{18.1}_{2.3}$ | $9.2^{17.5}_{1.7}$ | $6.3^{7.5}_{5.7}$ | $6.3^{11.5}_{5.2}$ |

Table 9.3: Summary table for the comparison between SWF levels 0 and $\tilde{0}$ and Ambisonics order 1 and 2, decoded to a 7.0.4 layout with the focus preset. Each entry reports the average and maximum and minimum values of the specified observable, $\mathrm{avg}^{\mathrm{max}}_{\mathrm{min}}$. Highlighted in italic the values of mean radial intensity that are similar across different formats. Highlighted in blod the highest value for the mean radial intensity.

| observable | VBAP-SWF 1 | OPT-SWF 1 | Ambi 2 | Ambi 3 |
|---|---|---|---|---|
| $E$ energy (dB) | $0.00^{0.43}_{-0.23}$ | $-0.10^{0.32}_{-0.46}$ | $-0.06^{0.03}_{-0.15}$ | $0.01^{0.14}_{-0.12}$ |
| $I_{\mathrm{R}}$ intensity | $\mathit{0.79}^{0.86}_{0.63}$ | $\mathit{0.79}^{0.87}_{0.63}$ | $\mathit{0.81}^{0.89}_{0.60}$ | $\mathbf{0.86}^{0.92}_{0.63}$ |
| $I_{\mathrm{T}}$ intensity | $0.09^{0.16}_{0.05}$ | $0.10^{0.22}_{0.01}$ | $0.11^{0.20}_{0.09}$ | $0.08^{0.17}_{0.03}$ |
| $I_{\mathrm{T}}$ intensity (deg) | $5.2^{9.2}_{2.9}$ | $5.7^{12.7}_{0.6}$ | $6.3^{11.5}_{5.2}$ | $4.6^{9.8}_{1.7}$ |

Table 9.4: Summary table for the comparison between SWF level 1 and Ambisonics order 1 and 2, decoded to a 7.0.4 layout with the smooth preset. Each entry reports the average and maximum and minimum values of the specified observable, $\mathrm{avg}^{\mathrm{max}}_{\mathrm{min}}$. Highlighted in italic the values of mean radial intensity that are similar across different formats. Highlighted in blod the highest value for the mean radial intensity.



| observable | VBAP-SWF 1 | OPT-SWF 1 | Ambi 2 | Ambi 3 |
|---|---|---|---|---|
| $E$ energy (dB) | $-0.12^{0.74}_{-0.70}$ | $-0.24^{0.59}_{-1.08}$ | $-0.06^{0.03}_{-0.15}$ | $0.01^{0.14}_{-0.12}$ |
| $I_{\mathrm{R}}$ intensity | $\mathbf{0.87}^{0.96}_{0.63}$ | $\mathbf{0.87}^{0.96}_{0.64}$ | $0.81^{0.89}_{0.60}$ | $\mathbf{0.86}^{0.92}_{0.63}$ |
| $I_{\mathrm{T}}$ intensity | $0.10^{0.29}_{0.02}$ | $0.11^{0.26}_{0.00}$ | $0.11^{0.20}_{0.09}$ | $0.08^{0.17}_{0.03}$ |
| $I_{\mathrm{T}}$ intensity (deg) | $5.7^{16.9}_{1.1}$ | $6.3^{15.1}_{0.0}$ | $6.3^{11.5}_{5.2}$ | $4.6^{9.8}_{1.7}$ |

Table 9.5: Summary table for the comparison between SWF level 1 and Ambisonics order 1 and 2, decoded to a 7.0.4 layout with the focus preset. Each entry reports the average and maximum and minimum values of the specified observable, $\mathrm{avg}^{\mathrm{max}}_{\mathrm{min}}$. In bold we highlight the highest values of mean radial intensity, which in this case are also similar across different formats.

listenings. This supports the fact that the radial intensity models well the perceived source size. With the tests we performed, it is difficult to quantify how relevant is the small non-zero transverse intensity for the incorrect positioning of the source. Specific tests should be carried out. As for the loudness across the horizontal trajectory of the audio source, we didn't notice any variation with these moderate energy differences.

## 9.2 Objective Evaluation of OPT-SWF and Ambisonics for the Hamasaki 22.2 Layout

In this Section we would like to briefly illustrate a reduced comparison between OPT-SWF and Ambisonics for a layout that is the channel-based setup with the most number of speakers to date, the Hamasaki 22.2 [Hamasaki et al., 2005]. For this comparison we are not in the position to be able to assess the differences also via listening tests, even if limited and informal. We rely only on the psychoacoustic indicators described in the manuscript. For this comparison, we created a decoding for OPT-SWF at level 1 and Ambisonics at order 3. In both cases the number of loudspeakers exceeds the number of format's channels to decode.

The Hamasaki 22.2 layout is composed of 22 speakers disposed in three levels and 2 subwoofers. The lower level has 3 loudspeakers in the frontal area (between $-15^\circ$ and $-25^\circ$ of elevation). The middle layer has 10 loudspeakers, and it is possible to think to it as an enriched 9.0 layout with an added speaker right in the back of the sweet spot (or opposite to the center channel). The top layer (between $30^\circ$ and $45^\circ$ of elevation) has 8 loudspeakers at an equal angular distance between each other. To reach the number of 22 loudspeakers, the last loudspeaker is placed at the zenith of the layout ($90^\circ$ elevation), typically called "voice of god".

In Figures 9.25 and 9.26 we report the panning functions for an horizontal panning for Ambisonics at order 3 and OPT-SWF at level 1 decoded to the Hamasaki 22.0 layout described. The solid lines represent the gains of the speakers located in the horizontal plane (zero elevation) while the dashed lines depict the gains of the speakers with non-zero elevation. This representation

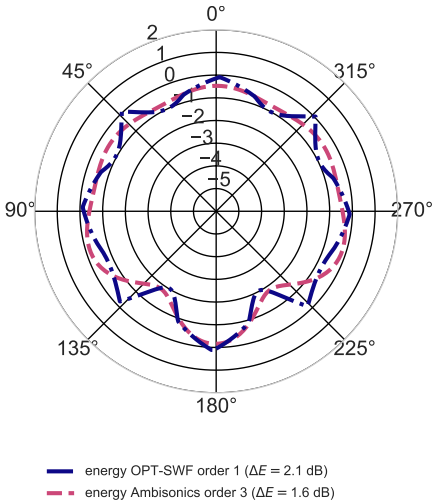



Figure 9.23: Energy comparison comparison for OPT-SWF at level 1 and Ambisonics at order 3, decoded to an Hamasaki 22.0 layout, on the horizontal plane.

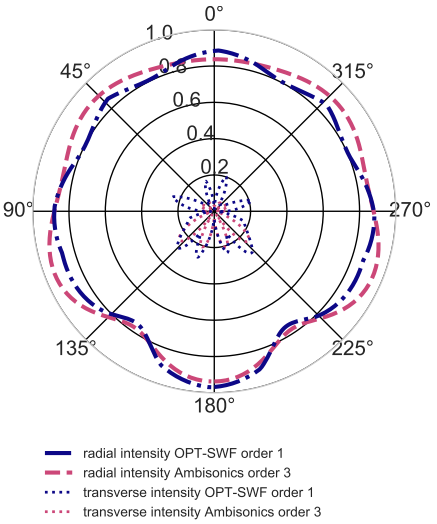



Figure 9.24: Intensity comparison comparison for OPT-SWF at level 1 and Ambisonics at order 3, decoded to an Hamasaki 22.0 layout, on the horizontal plane. The dashed lines represent radial intensity and the dotted ones the transverse intensity component.

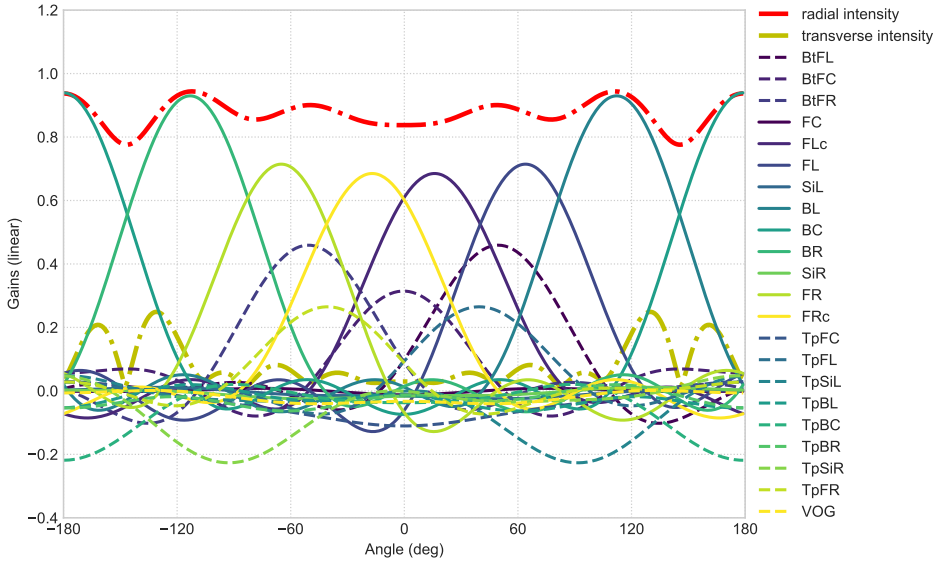



Figure 9.25: Horizontal panning for Ambisonics at order 3 decoded to an Hamasaki 22.0 layout.

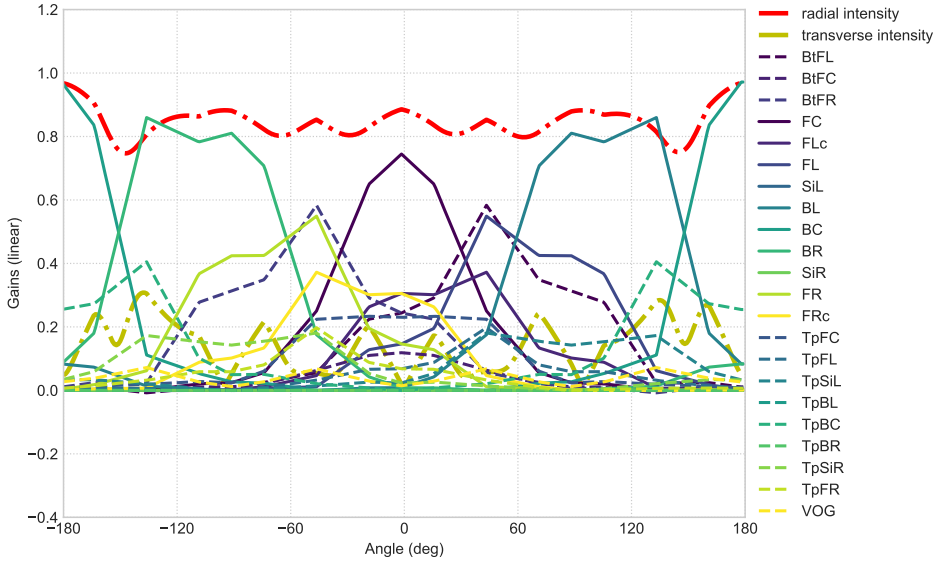



Figure 9.26: Horizontal panning for OPT-SWF at level 1, decoded to an Hamasaki 22.0 layout.

makes apparent that some speakers in the top layer are activated even if the panning is purely in the horizontal plane, that overlaps with the Hamasaki's middle layer. The number of speakers activated by Ambisonics and OPT-SWF is very similar. The only difference, that in this specific case we consider minimal, is the type of speakers activated: for a source located in the center in OPT-SWF the central speaker is activated while in Ambisonics the left and right speakers generate a phantom center.

In Figures 9.23 and 9.24 depict on the horizontal plane the reconstructed energy and intensity, respectively, for Ambisonics at order 3 and OPT-SWF at level 1 decoded to the Hamasaki 22.0. It is possible to see that both techniques achieve similar performances. The $\Delta E$ is limited around 1 dB for both techniques except to the left and right of the back speaker (BC). That is an effect due to the distance between the speaker at $\pm 110^\circ$ and the speaker at $180^\circ$, and is controlled by a (tunable) parameter of IDHOA. We will elaborate more in the discussion about the vertical plane plots.

A relevant difference that is not shown in the horizontal plots is the actual difficulty to generate an Ambisonics decoding with very limited negative gains. The tweaking of parameters for the minimization parameters is non trivial, especially for layouts that cover only partially the surface of the sphere. This is probably one of the reasons why there are no commercial tools to generate Ambisonics decoding matrices, but typically in commercial products static matrices are provided. For OPT-SWF essentially any decoder produced will have very limited negative gains by construction. To stress this point, we report the same plots, gains, energy and intensity, for the vertical plane.

In Figures 9.29 and 9.30 we report the panning functions for a panning around the vertical plane for Ambisonics at order 3 and OPT-SWF at level 1 decoded to the Hamasaki 22.0 layout described. It is apparent that Ambisonics struggles to maintain positive gains, while for OPT-SWF it is guaranteed by the filters themselves. To try to not compensate for the areas without loudspeakers, i.e. the lower half of the sphere, in IDHOA we implemented a mechanism to reduce the contribution of these areas in the value of the cost function. This is the motivation for the drop in the reconstructed energy between the $180^\circ$ and $315^\circ$ elevation, see Figure 9.27. The same happens for the radial intensity, Figure 9.28.

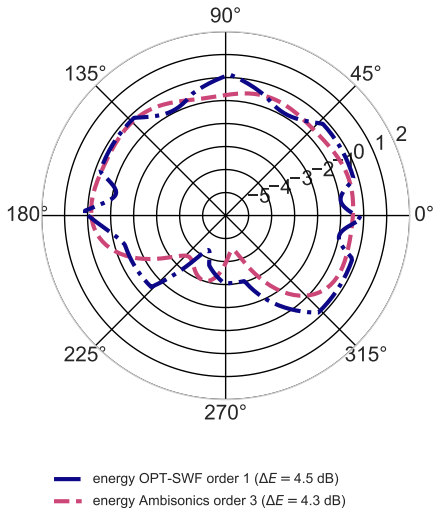



Figure 9.27: Energy comparison comparison for OPT-SWF at level 1 and Ambisonics at order 3, decoded to an Hamasaki 22.2 layout, on the vertical plane.

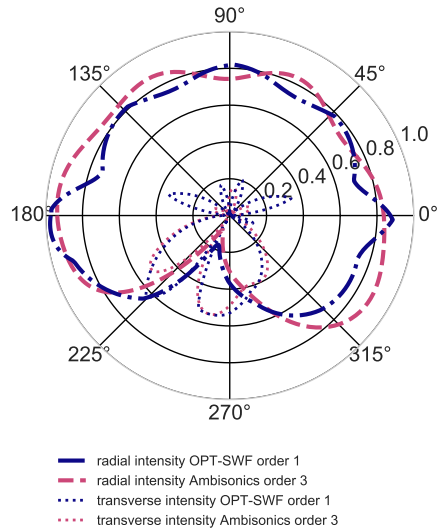



Figure 9.28: Intensity comparison comparison for OPT-SWF at level 1 and Ambisonics at order 3, decoded to an Hamasaki 22.2 layout, on the vertical plane. The dashed lines represent radial intensity and the dotted ones the transverse intensity component.

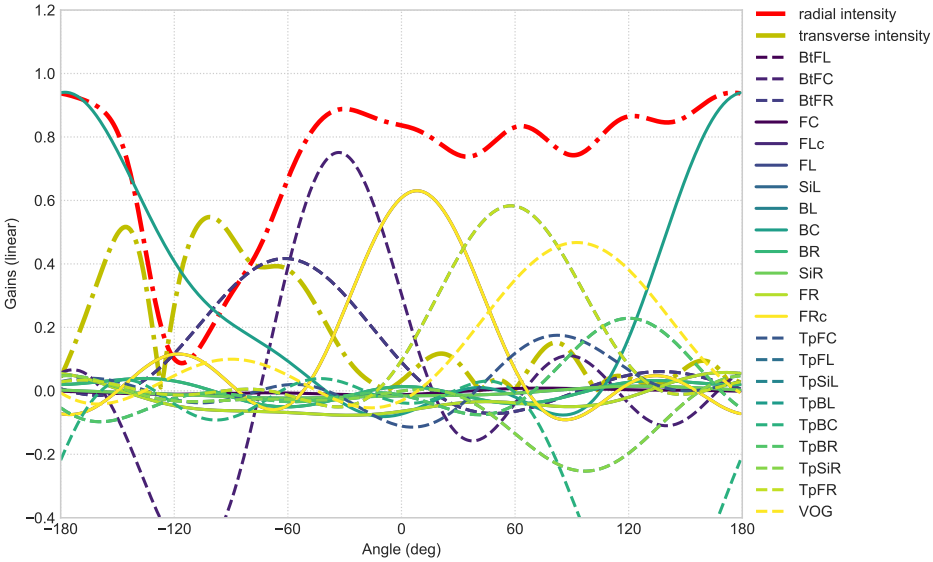



Figure 9.29: Vertical panning for Ambisonics at order 3 decoded to an Hamasaki 22.2 layout.

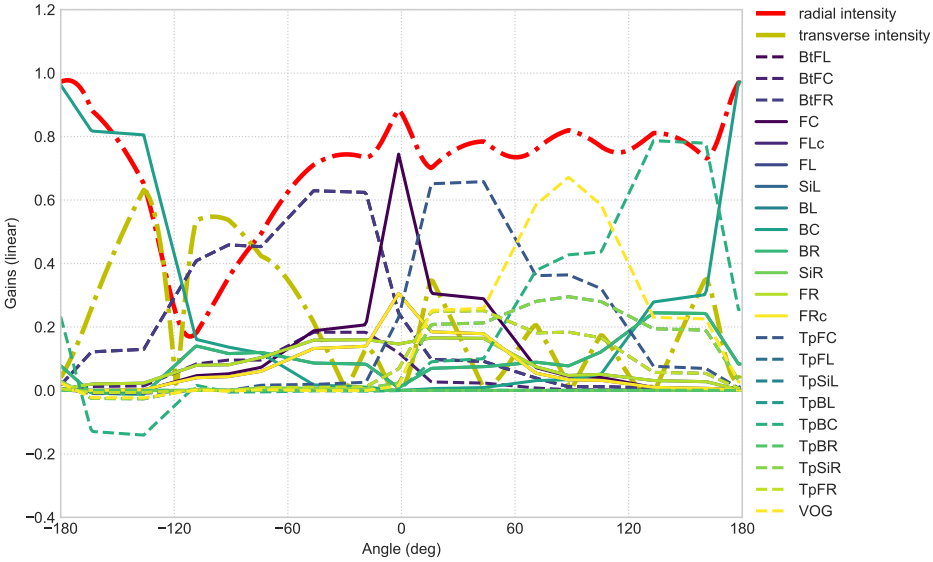



Figure 9.30: Vertical panning for OPT-SWF at level 1, decoded to an Hamasaki 22.2 layout.

Both Ambisonics and the SWF implementation we show here have problems dealing with irregularly spaced speakers. It is quite evident in the 22.2 layout, since the layout is hemispherical and the frontal area is more dense in speakers than the rear zone. The Hamasaki distribution of speakers is more concentrated where the human perception accuracy is greater: in the horizontal plane and especially in the frontal area. Both ambisonics and (this implementation of) SWF treat all directions equally: Ambisonics because of the support of the SH and SWF because our mesh has (almost) equispaced points on the sphere. The improvement that SWF brings is the easiness of producing a decoding with positive gains. Another implementation of SWF with an irregular sampling by design (irregular mesh) could deal with irregular layouts natively.

## 9.3 Summary

The SWF implementation described in Chapter 8.1 is designed to be compared directly with Ambisonics: covers the full sphere, has equispaced sampling points that correspond to one of the platonic solids (octahedron or 3-design) to which Ambisonics has closed form decoding equations. In this Chapter we compared the decoding of SWF and Ambisonics to two irregular layouts of speakers. TWe summarize this comparison with a list of advantages and disadvantages of SWF decoding over Ambisonics decoding:

+ More control of negative gains. Implies bigger sweet spot and more robust imaging.

+ More predictable fine tuning. Implies that we can push the decoder in the *smooth* or *focus* direction without adding significant out-of-phase contributions.

+ Possibly more directional for a similar number of channels (for the *focus* decoder).

- The energy reconstruction is more irregular. It is the price to pay for not having negative gains and encoding with the requirement of preserving pressure.

- At level 1 is mostly equivalent to Ambisonics at order 3. Because of the construction of SFW along the lines of Ambisonics, the two techniques are not significantly different.

If we look at the performance in terms of radial intensity, the two formats perfom similarly for an irregular array of speakers. Ambisonics stands out for the smooth energy reconstruction, while SWF for the absence of negative gains and decoding flexibility.

# Chapter 10

# CONCLUSIONS AND FUTURE WORK

## 10.1 Conclusions and Discussion

In this thesis we have defined a new generic framework for spatial audio encodings based on wavelet filters. We have described the complete audio workflow that makes use of this new tool. Then, we have particularized the framework to the spherical case for a specific mesh construction, resulting in a practical realization: the spherical wavelet format (SWF). Similarly to Ambisonics, this format is based on channels, but in the case of SWF the channels have a particular spatial localization. On the encoding side of the audio chain, we have devised a numerical method for wavelets optimization (with short filter length), enabling the creation of a possibly infinite set of core filters. On the decoding side of the audio chain, we have built and made publicly available[1] a universal decoding method, based on the numerical optimization of some psychoacoustical observables.

**SWF** The objective of this thesis was to build a new channel agnostic format, that is homogeneous and coherent, but also has good localization with

[1] The version of IDHOA used in this work will be made public soon.

few channels, easily handles irregular layouts and holds well when moving out of the sweet spot. We have depicted the full audio chain: encoding to mesh and downsampling, transmission, upsampling and decoding to speakers. The new format is effectively channel agnostic, there is no reference to the destination layout in the definition of the format. The homogeneity and coherence characteristics need a more detailed discussion.

SWF is homogeneous in the specific implementation defined in the thesis, since the mesh over which is defined the format is in fact homogeneous. Nevertheless, it could be perfectly possible to define a wavelet format with a non-homogeneous mesh and obtain a non-homogeneous format. The same in fact happens with Ambisonics when decoding to irregular layouts. Ambisonics is by construction homogeneous, and the decoders for regular layouts are also (typically) homogeneous. However, when decoding to irregular layouts the theory does not assure that the resulting decoding will be homogeneous. Actually, the best decoders for irregular layouts are not homogeneous (like the ones produced by IDHOA and presented in this thesis). In this case SWF and Ambisonics are no different. The coherence follows a similar train of thought, both SWF and Ambisonics can be coherent in special conditions. SWF has indeed good localization with few channels and, thanks to the extremely limited negative gains, holds well when moving out of the sweet spot, like an amplitude panner does. It has been demonstrated that SWF behaves well when decoding to layouts that are irregular (in the SWF sense) and with the help of IDHOA it is possible to generate meaningful decoders in a matter of minutes.

We have explored three variations of a particular incarnation of this format. In both cases wavelets were defined over a spherical mesh, created from a primitive solid (an octahedron) using a Loop subdivision scheme. In the first variation the wavelet family is implicitly defined by the VBAP panning rule. In the second variation we use an off-the-shelf wavelet family, called interpolating wavelet. In the third variation the wavelets were optimized numerically by a brute-force method. The three methods generate audio formats that have very similar characteristics in terms of energy and intensity reconstruction. The main differences lay in the shape of the panning functions and in the behaviour of the upsampling matrix, $\mathbf{P}$. It is to be noted that the three examples explored do not necessarily represent the best possible realizations. One of the virtues

of SWF is precisely the ability to adapt to many different situations. We believe that there is no such thing as the best possible SWF format, but rather it depends on the particular context and goal. One of the drawbacks of SWF with respect to Ambisonics is that the acoustical and perceptual interpretation of the format in terms of pressure and velocity is lost in general (we still retain the notion of the global pressure by a careful wavelet design). In this context, it is key to have a acoustically and perceptually motivated decoder that can reinstate the missing physical and perceptual observables.

A three-element comparison, OPT-SWF (using an optimized wavelet), VBAP-SWF (trivial wavelet from VBAP) and Ambisonics, has been carried out for two different speakers layouts. Observations from reconstructed signals, reconstructed energy and intensity indicate that SWF is a format that, depending on the decoding, can fit between an amplitude panner and Ambisonics. It has localization characteristics similar to (or in some cases better than) Ambisonics, with greater control on the negative gains. Informal listening tests confirm these characteristics.

In our experience, the difference between the two variations of the wavelet format explored are relatively minor when evaluated in terms of the decoding results. We noticed that final results depend only slightly on the wavelet family as long as this family has been designed with reasonable characteristics. A possible explanation is that the IDHOA decoding minimizes any possible intrinsic differences between the different encodings. Also, notice that we have only explored meshes of relatively low order. It is possible that differences become more apparent when going to higher order meshes, since the filtering effects are cumulative. Additionally, besides the decoding characteristics, other filter design properties, e.g. encoding performance, can be considered when designing and evaluating the wavelet families. Anyway, we expect that the different characteristics of the wavelet families will be more evident when using custom subdivision meshes that represent directly standard speaker positions. When building custom meshes, the requirement for a spherical format could be lifted, and we could define a format for meshes with a non-spherical topology (e.g. half dome).

As a related remark, notice that the only filter which is strictly essential in our framework is the analysis filter $\mathbf{A}$, given that the decoding to speakers is

computed separately with IDHOA via a numerical optimization. However, we still believe that it is important to have a complete wavelet representation. When optimizing the filters it is important to optimize at the same time the analysis and synthesis filters **A** and **P** to ensure that the wavelet transform can have a well behaved reconstruction. This fact can later on ease the task of decoding to speakers performed by IDHOA. Besides, on practical grounds, having a complete wavelet construction can be useful to be able to manipulate the spatial signal; on theoretical grounds, the wavelet construction is key to understand what is left out by the truncation of SWF, in the spirit of the sampling theorem, something we leave for future work.

Overall, SWF's encoding, transmission and decoding flexibility and rendering performance make it an interesting family of formats to explore in real-life conditions.

**IDHOA** A fundamental piece for the new format, and also for the comparison with the reference technology Ambisonics, is the stage of decoding to a real world layout. One of the main outputs of the work is the formulation and implementation of the IDHOA decoder. While initially oriented to solve the problem of decoding Ambisonics to irregular layouts (to date still relevant), we developed an algorithm that, leveraging psychoacoustic criteria, can generate a decoding matrix for any linear encoding format, as long as this encoding format allows encoding a point source in any direction. The main novelty factor is the separation of intensity vectors in radial and tangential components, making it possible to optimize the two components separately.

Often in the past literature there has been an excessive stress, in our opinion, on reducing the tangential component of velocity and intensity. This limits the possibility of the radial part to reach relevant values, thus effectively broadening the audio sources and making them difficult to localize in space. In the same line of thought, forcing the velocity or intensity vector to have the same value along the area covered by loudspeakers, which is the homogeneity characteristic of Ambisonics, does not make much sense for irregular layouts. We think that separating the velocity and intensity into their radial and tangent part, and trying to maximize the radial component without trying to make it uniform, while minimizing the tangential one without requiring it to

be strictly zero, generates decodings that are in practice much superior to the commonly available ones. Along with the IDHOA code we will publish all the decoding matrices used in this thesis to decode Ambisonics to the mentioned layouts.

One final note, we believe that adopting this factorization in components for the velocity and intensity vectors should become a common practice among the researchers in the area of Ambisonics decoding especially for reporting the results for different decoders and technologies. Typically the results are reported with obscure sphere projections and reporting only the modulus of the vectors. We think that it would be useful to adopt a common format for reporting, that has proven to be very immediate to relate to the actual listening experience.

## 10.2 Future Work

Reach out to the community disseminating this work is indeed our first priority. From there, we hope to get some feedback and shape the upcoming goals. Ideally, it would be interesting to design a more industry-oriented format, that does not necessarily need to compare directly with Ambisonics.

Along the line of searching for a more industry-oriented format and in the spirit of (compressing) wavelets, whose philosophy is "to model a function, use a function similar to the function you want to model", it would be interesting to experiment with meshes similar to the destination speakers' layout. It should be quite straightforward to test this concept on any mesh using a VBAP-SWF, while generating an optimized wavelet format requires more work and fine-tuning.

Given the flexibility of our construction, a possibly interesting application to test would be using SWF as a spatial encoder for other formats. For example, it is possible to decode Ambisonics to a SWF mesh with a basic decoding (pressure preserving), then manipulate the signals via the described SWF tools (if needed), and decode perceptually with IDHOA to the destination layout. Note that the spatial operations like rotations, zoom or spatial deformation are quite trivial over the $\mathbf{c}$ signals, since they have a spatial meaning in the three-dimensional space, and the usual operations apply. Another example, SWF

could be used as a transport format for object-based formats to reduce the number of transmitted audio files. The object based format could be encoded to SWF, manipulated and transmitted, and decoded to the loudspeakers' layout via IDHOA.

After format and specific wavelet definition it would be interesting to understand what is left out by the truncation of SWF, i.e. the meaning of the 'details', in the spirit of the sampling theorem.

# Bibliography

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Abdi, 2010] Abdi, H. (2010). Holm's sequential Bonferroni procedure. In Salkind, N. J., editor, *Encyclopedia of research design*, pages 1–8. Sage Thousand Oaks, CA.

[Adriansen, 2015] Adriansen, F. (2015). AmbDec.

[Arfken et al., 2005] Arfken, G. B., Weber, H. J., and Harris, F. E. (2005). *Mathematical Methods for Physicists, Sixth Edition: A Comprehensive Guide*. Academic Press, 6 edition.

[Arteaga, 2013] Arteaga, D. (2013). An Ambisonics decoder for irregular 3-D loudspeaker arrays. In *Audio Engineering Society Convention 134*.

[Barford et al., 1992] Barford, L. A., Fazzio, R. S., and Smith, D. R. (1992). An introduction to wavelets. Technical report, Hewlett Packard.

[Bates, 2009] Bates, E. (2009). *The composition and performance of spatial music*. PhD thesis.

[Batke and Keiler, 2010] Batke, J.-M. and Keiler, F. (2010). Using VBAP-derived panning functions for 3d Ambisonics decoding. In *2nd Ambisonics Symposium, Paris*.

[Beezer, 2012] Beezer, R. A. (2012). Reduced row-echelon form.

[Benjamin et al., 2010] Benjamin, E., Heller, A., and Lee, R. (2010). Design of Ambisonic decoders for irregular arrays of loudspeakers by non-linear optimization. In *Audio Engineering Society Convention 129*.

[Berkhout, 1988] Berkhout, A. J. (1988). A holographic approach to acoustic control. *J. Audio Eng. Soc*, 36(12):977–995.

[Boehm, 2011] Boehm, J. (2011). Decoding for 3-D. In *Audio Engineering Society Convention 130*.

[Bureau, 1997] Bureau, U. S. C. (1997). Geographic information systems FAQ.

[Catmull and Clark, 1978] Catmull, E. and Clark, J. (1978). Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355.

[Daniel, 2000] Daniel, J. (2000). *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*. PhD thesis, Université Paris 6.

[Daniel, 2003] Daniel, J. (2003). Spatial sound encoding including near field effect: Introducing distance coding filters and a viable, new ambisonic format. In *Audio Engineering Society Conference: 23rd International Conference: Signal Processing in Audio Recording and Reproduction*.

[Daniel et al., 2003] Daniel, J., Moreau, S., and Nicol, R. (2003). Further investigations of high-order ambisonics and wavefield synthesis for holophonic sound imaging. In *Audio Engineering Society Convention 114*. Audio Engineering Society.

[Daubechies, 1990] Daubechies, I. (1990). The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5):961–1005.

[Daubechies, 1992] Daubechies, I. (1992). *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics.

[Doo, 1978] Doo, D. (1978). A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In *Proceedings of the International Conference on Computer-Aided Design*, pages 157–165. IEEE Computer Society.

[Dremin et al., 2001] Dremin, I. M., Ivanov, O. V., and Nechitailo, V. A. (2001). Wavelets and their use. *Phys. Usp.*, 44:447.

[Frank, 2013] Frank, M. (2013). *Phantom Sources using Multiple Loudspeakers in the Horizontal Plane*. PhD thesis, IEM Graz, Austria.

[Geiger, 2018] Geiger, M. (2018). Compute jacobian and hessian with pytorch.

[Gerzon and Barton, 1992] Gerzon, M. A. and Barton, G. J. (1992). Ambisonic decoders for HDTV. In *Audio Engineering Society Convention 92*.

[Hamasaki et al., 2005] Hamasaki, K., Hiyama, K., and Okumura, R. (2005). The 22.2 multichannel sound system and its application. In *Audio Engineering Society Convention 118*.

[Heller et al., 2012] Heller, A., Benjamin, E., and Lee, R. (2012). A toolkit for the design of ambisonic decoders. In *Linux Audio Conference*.

[ITU-R, Recommendation BS, 2003] ITU-R, Recommendation BS (2003). 1534-1,'Method for the subjective assessment of intermediate quality levels of coding systems (MUSHRA)'.

[Johnson, 2007] Johnson, S. G. (2007). Nlopt library.

[Kammoun et al., 2012] Kammoun, A., Payan, F., and Antonini, M. (2012). Sparsity-based optimization of two lifting-based wavelet transforms for semi-regular mesh compression. *Computers & Graphics*, 36(4):272–282.

[Loop, 1987] Loop, C. (1987). *Smooth Subdivision Surfaces Based on Triangles*. phdthesis.

[Lossius et al., 2009] Lossius, T., Baltazar, P., and de la Hogue, T. (2009). Dbap - distance-based amplitude panning. In *ICMC*.

[Malham, 1999] Malham, D. (1999). Homogeneous and non-honogeneous surround sound systems. In *Audio Engineering Society Conference: UK 14th Conference: Audio - The Second Century*.

[Mallat, 1989] Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693.

[mh acoustics LLC, 2019] mh acoustics LLC (2019). The eigenmike microphone array. https://mhacoustics.com/products.

[Moore and Wakefield, 2007] Moore, D. and Wakefield, J. (2007). The design and analysis of first order ambisonic decoders for the ITU layout. In *Audio Engineering Society Convention 122*.

[Moore and Wakefield, 2011] Moore, D. and Wakefield, J. (2011). Designing ambisonic decoders for improved surround sound playback in constrained listening spaces. In *Audio Engineering Society Convention 130*.

[Olsen et al., 2007] Olsen, L., Samavati, F., and Bartels, R. (2007). Multiresolution for curves and surfaces based on constraining wavelets. *Computers & Graphics*, 31(3):449–462.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

[Poletti, 2005] Poletti, M. A. (2005). Three-dimensional surround sound systems based on spherical harmonics. *J. Audio Eng. Soc*, 53(11):1004–1025.

[Pulkki, 1997] Pulkki, V. (1997). Virtual sound source positioning using vector base amplitude panning. *J. Audio Eng. Soc*, 45(6):456–466.

[Rowan, 1990] Rowan, T. H. (1990). *Functional Stability Analysis of Numerical Algorithms*. PhD thesis, Austin, TX, USA. UMI Order No. GAX90-31702.

[Scaini, 2013] Scaini, D. (2013). Idhoa, software for decoding of High Order Ambisonics to irregular layouts. https://github.com/BarcelonaMedia-Audio/idhoa.

[Scaini, 2015] Scaini, D. (2015). Idhoa, software for decoding of High Order Ambisonics to irregular layouts. https://github.com/davrandom/idhoa.

[Scaini and Arteaga, 2014] Scaini, D. and Arteaga, D. (2014). Decoding of higher order ambisonics to irregular periphonic loudspeaker arrays. In *Audio Engineering Society Conference: 55th International Conference: Spatial Audio*.

[Scaini and Arteaga, 2015] Scaini, D. and Arteaga, D. (2015). An evaluation of the IDHOA Ambisonics decoder in irregular planar layouts. In *Audio Engineering Society Convention 138*.

[Scaini and Arteaga, 2019a] Scaini, D. and Arteaga, D. (2019a). In preparation.

[Scaini and Arteaga, 2019b] Scaini, D. and Arteaga, D. (2019b). Wavelet based spherical audio format. Submitted.

[Schmele et al., 2013] Schmele, T., García-Garzón, D., Sayin, U., Scaini, D., and Arteaga, D. (2013). Layout remapping tool for multichannel audio productions. In *Audio Engineering Society Convention 134*.

[Schröder and Sweldens, 1995] Schröder, P. and Sweldens, W. (1995). Spherical wavelets: efficiently representing functions on a sphere. In *Wavelets in the Geosciences*, pages 158–188. Springer-Verlag.

[Stollnitz et al., 1995] Stollnitz, E. J., Derose, T. D., and Salesin, D. H. (1995). Wavelets for computer graphics: A primer - part 2. *IEEE Computer Graphics and Applications*, 15(4):75–85.

[Sweldens, 1998] Sweldens, W. (1998). The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511–546.

[Sweldens and Schröder, 1995] Sweldens, W. and Schröder, P. (1995). Building your own wavelets at home.

[Theano Development Team, 2016] Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

[Tsang and Cheung, 2009] Tsang, P. and Cheung, K. (2009). Development of a re-configurable ambisonic decoder for irregular loudspeaker configuration. *Circuits, Devices & Systems, IET*, 3(4):197–203.

[van Rossum, 1995] van Rossum, G. (1995). Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam.

[Wächter and Biegler, 2006] Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

[Walter and Lehmann, 2013] Walter, S. F. and Lehmann, L. (2013). Algorithmic differentiation in python with algopy. *Journal of Computational Science*, 4(5):334 – 344.

[Wickerhauser, 1994] Wickerhauser, M. (1994). Wavelets: Algorithms and applications (yves meyer). *SIAM Review*, 36(3):526–528.

[Wiggins, 2004] Wiggins, B. (2004). *An investigation into the real-time manipulation and control of three-dimensional sound fields*. PhD thesis, University of Derby.

[Wiggins et al., 2003] Wiggins, B., Paterson-Stephens, I., Lowndes, V., and Berry, S. (2003). The design and optimisation of surround sound decoders using heuristic methods. In *Proceedings of UKSim*, volume 3, pages 106–114.

[Xu, 2011] Xu, E. (2011). PyIpopt, BSD license.

[Zotter and Frank, 2012] Zotter, F. and Frank, M. (2012). All-round ambisonic panning and decoding. *J. Audio Eng. Soc*, 60(10):807–820.

[Zotter and Frank, 2018] Zotter, F. and Frank, M. (2018). Ambisonic decoding with panning-invariant loudness on small layouts (allrad2). In *Audio Engineering Society Convention 144*.

[Zotter and Frank, 2019] Zotter, F. and Frank, M. (2019). *Ambisonics*. Springer Open.

[Zotter et al., 2012] Zotter, F., Pomberger, H., and Noisternig, M. (2012). Energy-preserving ambisonic decoding. *Acta Acustica united with Acustica*, 98(1):37–47.

[Zylia, 2019] Zylia (2019). https://www.zylia.co/products.html.

# Appendix A

# MINIMIZATION PROBLEMS IN PYTHON VIA IPOPT AND PYTORCH

Both the optimization of the decoding cost function and the wavelets one are the result of several years trying and experimenting with different technologies. We feel that it is relevant to leave in writing if not the whole trajectory, at least the sketch of the final implementation. During the last 7 years we tried NLopt [Johnson, 2007], IPOPT [Wächter and Biegler, 2006] as minimization libraries, and several libraries for auto-differentiation, namely algopy [Walter and Lehmann, 2013], Theano [Theano Development Team, 2016], Tensorflow [Abadi et al., 2015] and PyTorch [Paszke et al., 2017]. The final implementation uses IPOPT over NLopt, and PyTorch over the rest of mentioned libraries. This combination is the one that proved to have the fastest execution times, ease of debugging and programming flexibility.

## A.1 IPOPT Minimization Library

IPOPT (Interior Point OPTimizer) is an open-source software package for large non-linear optimization. It can be used to solve general nonlinear programming problems, including arbitrary constraints. The software itself is

written in C++ but has several native or contributed APIs for other languages. In our case, we wanted to interface with Python and we used PyIpopt [Xu, 2011] as the Python API for IPOPT. IPOPT requires the computation of the Jacobian of the cost function and the constraints. If provided, IPOPT uses also the Hessian of both cost function and constraints, otherwise it will internally calculate it numerically.

The main obstacle is then calculating the first and second derivatives. IPOPT just requires the numerical value of the derivatives, so the choice of the method is left to the researcher.

## A.2 Calculating Derivatives

There are essentially three methods available, with their benefits and drawbacks. We will list them schematically in the following:

1. Analytic differentiation: derivatives are computed and implemented once, by hand or with the help of some computer algebra software,

    + Exact derivatives, the numerical evaluation is fast.
    - The process of (manually) calculating the derivatives is time consuming, the implementation can be very complicated.
    - Not flexible: the derivatives have to be recalculated at every change in cost function or constraints.

2. Numerical differentiation: approximate the derivatives by finite differences

    + Can be always calculated, even when the cost function doesn not have a closed analytic expression.
    - Approximation errors arise (round-off and truncation) and they do accumulate.
    - The evaluation can be really slow.

3. Automatic differentiation

+ Exact derivatives, once calculated the numerical evaluation is fast.
+ Very flexible: since the derivatives are exact and calculated automatically, it is possible to experiment with cost function and constraints.
- Can be calculated only if the cost function or constraints can be expressed in terms of operations whose derivative is known by the library.
- Needs some adaptation of the algorithms and introduces an external dependancy.
- Can be slow of fast depending on the specific library.

We chose the analytic differentiation, since we want the flexibility to experiment with different cost functions and constraints. In the recent years packages for automatic differentiation have evolved dramatically thanks to the rise of artificial intelligence.

### A.2.1 Automatic Differentiation Packages

Generally, the automatic differentiation packages are built on the fact that the the derivative of any expression can be computed using the chain rule. Applying several times the chain rule, the composite expression is broken in elementary operations and functions whose derivatives are known. This way, the derivative of the initial function is computed algorithmically in a finite number of steps. The automatic differentiation software has to build an internal representation of the derivative of the function, which is called computational graph. This computational graph can be built statically (first build the graph, then compile it) of dynamically (the graph is built at execution time). Different libraries tend to use different methods, even if lately this difference is gradually being relaxed.

We initially started our research using Theano, but we hit a wall when starting to use sparse matrices (for reducing the problem's dimensionality) and Theano team announced that they would cease the development (3 October 2017). We then moved to Tensorflow, that at the time built the computational

graph only statically. A part from the inherent difficulty of debugging (cryptic error messages that are related to internal graph and not the actual code), the code results difficult to reuse for interfacing with IPOPT. Ideally we would like to have the same function that outputs numerical values for IPOT and that is the input to the automatic differentiation algorithm. We found the interaction with IPOPT much more neat to handle using PyTorch.

### A.2.2 Examples of Automatic Differentiation in Tensorflow and PyTorch

In the following, we will present a simple example showing how to calculate the derivative of a function in the statical paradigm (with Tensorflow) and in the dynamic paradigm (with PyTorch). Starting with Tensorflow:

```
# derivatives of a function in tensorflow

import tensorflow as tf

# get a number from terminal
print("type a number and press enter")
point = input()
point = float(point)
data = tf.placeholder(dtype=tf.float32, shape=())


# the function you want to calculate the gradient
def function(indata):
    square = tf.pow(indata, 2)
    return square


# start a TensorFlow session
# (you need it to evaluate numerically the symbolic expressions)
sess = tf.Session()

# TF symbolic version of function
tf_function = function(data)
# value of function in your point
```

```
value_function = tf_function.eval(feed_dict={data: point},
    session=sess)
print("Function value (the value you entered squared): %.3f" %
    value_function)

# calculate the symbolic gradient of function
grad_function = tf.gradients(tf_function, [data])[0]
# evaluate the gradient in your point
value_grad_funct = grad_function.eval(feed_dict={data: point},
    session=sess)
print("Derivative of the function's value (twice the value you
    entered): %.3f" % value_grad_funct)
```

The output of the code is:

```
type a number and press enter
3
Function value (the value you entered squared): 9.000
Derivative of the function's value (twice the value you entered):
    6.000
```

While with PyTorch:

```
# derivatives of a function in pytorch

import torch


# get a number from terminal
print("type a number and press enter")
point = input()
point = float(point)
data = torch.tensor(point, requires_grad=True,
    dtype=torch.float32)

# the function you want to calculate the gradient
def function(indata):
    square = torch.pow(indata, 2)
    return square

```

```
17
18 # value of function in your point (it's a torch object!)
19 value_function = function(data)
20 print("Function value (the value you entered squared): %.3f" %
   ↪  value_function)
21
22 # calculate the symbolic gradient of function in your point
23 value_function.backward()
24 # get the value of the gradient in your point
25 value_grad_funct = data.grad.numpy()
26 print("Derivative of the function's value (twice the value you
   ↪  entered): %.3f" % value_grad_funct)
```

The output is obviously the same as in the previous formulation. Even if we tried to maintain the same steps, it is apparent than in the PyTorch case the torch objects and the numerical values of those objects are carried together. This makes much easier the (numerical) debugging.

To scale this simple example to matrices and complex operations is trivial. Nevertheless there is a small detail missing that is very relevant in our context. When scaling the problem of calculating the derivatives of a function that accepts as input a matrix (all) the libraries for automatic differentiation used in deep learning return the sum of the derivatives, and not the full Jacobian. For this reason we have to calculate a derivative for each component of the input matrix, which is called *stride*. Fortunately, there is a small library that does exactly this striding for us [Geiger, 2018]. Calculating Jacobians and Hessians becomes extremely easy, as an example:

```
1 # jacobian and hessian of a function in tensorflow
2
3 import torch
4 import hessian as h
5
6 # define two variables
7 x = torch.tensor([1.5, 2.5], requires_grad=True)
8 y = torch.tensor([5.5, -4.], requires_grad=True)
9
10 # define the function
```

```
function = x.pow(y)
# calculate its jacobian
jac = h.jacobian(function, [x, y])
# print result
print("Jacobian")
print(jac)


# define the function
function = x.pow(2).prod().sum()
hes = h.hessian(function, x)
# print result
print("Hessian")
print(hes)
```

---

The output is:

```
Jacobian
tensor([[ 3.4101e+01, -0.0000e+00,  3.7710e+00,  0.0000e+00],
        [ 0.0000e+00, -4.0960e-02,  0.0000e+00,  2.3457e-02]])
Hessian
tensor([[12.5000, 15.0000],
        [15.0000,  4.5000]])
```

Having the autodifferentiation machinery sorted out, we can use PyIpopt [Xu, 2011] examples to start building a minimization using IPOPT as a minimization library.

# Appendix B

# STRATEGIES FOR PROBLEM DIMENSIONALITY REDUCTION

We have seen in Chapter 7 how the minimization problem for the wavelet filters is built, the cost functions we defined, and their constraints. In this Chapter we will analyze more in detail the (critical) implementation aspects of the wavelet optimization. Without loss of generality, we will focus on the first step of the minimization, the one where **A** and **P** are optimized together.

In the first step then the unknowns are the whole matrices **A** and **P**. The matrices connect two levels of the subdivision, so the number of elements in each of these two matrices is (number of mesh points at level n) $\times$ (number of mesh points at level n+1). To give some numbers, and referring to the mesh used for SWF (e.g. see Chapter 8), the $\mathbf{A}^1$ and $\mathbf{P}^1$ have $6 \times 18 = 108$ each, and the number of unknowns would be then $108 \times 2$. $\mathbf{A}^2$ has $18 \times 66 = 1188$ elements, and $\mathbf{A}^3$ has $66 \times 258 = 17028$. The dimensionality of the Jacobian and Hessian of the cost function grows with the growing number of variables (linearly for the Jacobian and quadratically for the Hessian). To this count, we have to add the constraints, in the case of the first step we have only $\mathbf{A}^j\mathbf{P}^j = \mathbf{1}$ (see Chapter 7), their Jacobian and Hessian. In the case of the already mentioned SWF, we have $6^2 = 36$ constraints at $j = 1$, $18^2 =$

$324$ at $j = 2$ and so on. Even at small dimension this brute-force approach, where all the elements of $\mathbf{A}$ and $\mathbf{P}$ are considered (to some extent) independent unknowns, tends to blow up quickly.

## B.1 Loop Spherical Subdivision Symmetries and Reduction of Degrees of Freedom

A part from the constraints we impose, $\mathbf{A}^j\mathbf{P}^j = \mathbf{1}$, we don't impose any particular symmetry and we treat the points of the mesh as if they were independent. Nevertheless, the mesh has its inherent symmetry which is given by the symmetry of the solid that we chose as initial mesh. The idea is then to explicitly impose the (original) mesh symmetry and get an effective reduction of the number of unknowns of the problem, resulting in a global reduction in dimensionality (less unknowns, less constraints, less derivatives).

(As already discussed, there are approaches where the optimization affects the prediction and update operators alone and not the full refinement matrices. In this case we wanted complete flexibility and possibly avoid the fractal shapes introduced by the recursion in the lifting together with the dual-lifting (see Subsection 5.4.5, and especially Eq. (5.22)).

With the help of a couple of figures we will illustrate the concept behind the reduction of the unknowns. In Figure B.1 we report the octahedral mesh (level 0) as seen from above. The visible vertices are the gray circles numbered in solid black $\{1,\ 2,\ 3,\ 4,\ 5\}$, the first four are positioned in the horizontal plane, while the number 5 is the vertex at the top. The vertex number 6 is the vertex at the bottom and is hidden in this figure, for this reason is indicated with a gray number. If we apply the Loop subdivision to this initial mesh, we obtain the mesh (at level 1) reported in Figure B.2 (the exact numbering of the vertices might not be the one reported in the figure, but it is not relevant in this context). We report with a dark red cross the new visible vertices, accompanied with their numbering. The vertices hidden by this type of projection are indicated in light red. The vertices added at this level are $\{7,\ 8,\ 9,\ 10,\ 11,\ 12, 13, 14, 15, 16, 17, 18\}$.

Now, if we take the vertex 1 at level 0 and search for its immediate neigh-

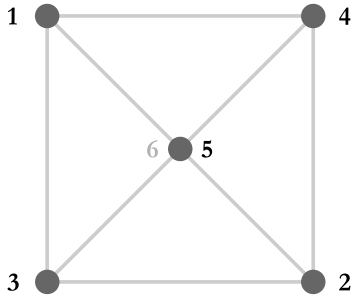



Figure B.1: View of the original octahedron from above. The visible vertices (dots) are numbered in solid black, the hidden vertex (6) is numbered in gray. The original mesh has 6 vertices.

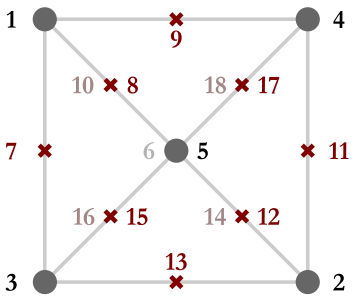



Figure B.2: View of the original octahedron with the first Loop subdivision. The original vertices are the dots, while the crosses represent the new verices produced by the Loop subdivision. The new visible vertices are numbered in solid dark red, while the remaining hidden ones are numbered in light red. The new mesh has in total 18 vertices.

bours at level 1, we will find the four vertices $\{7, 8, 9, 10\}$, and it is said that this vertex has valence 4. (The vertices at level 1 will have valence 6, when searching for their immediate neighbours at level 2. The Loop subdivision is a subdivision with valence 6.) We can schematically indicate this concept as:

$$\begin{array}{ccc} \text{level 0} & \text{has neighbours} & \text{level 1} \\ 1 & \rightsquigarrow & \{7, 8, 9, 10\} \end{array}$$

And we can write the same for the remaining vertices:

$$\begin{aligned} 2 &\rightsquigarrow \{11, 12, 13, 14\} \\ 3 &\rightsquigarrow \{13, 15, 7, 16\} \\ 4 &\rightsquigarrow \{9, 17, 11, 18\} \\ 5 &\rightsquigarrow \{8, 12, 15, 17\} \\ 6 &\rightsquigarrow \{10, 14, 16, 18\} \end{aligned}$$

Since the original mesh has rotational symmetry, we can build a function that maps each of the $\{2,\ 3,\ 4,\ 5,\ 6\}$ to the vertex number 1, and that maps their neighbours to 1's neighbours. Graphically, for the vertex 2:

$$\begin{array}{ccc} 2 & \rightsquigarrow & \{11,12,13,14\} \\ \downarrow & & \downarrow \\ 1 & \rightsquigarrow & \{7,8,9,10\} \end{array}$$

With this symmetry we reduce by 6 the number of unknowns (at the level of this example). Essentially the six rows of $\mathbf{A}$ (and the columns of $\mathbf{P}$) are shifted copies of each other.

Moreover, we can further reduce the dimensionality imposing left/right and up/down symmetries, this way the free parameters represented by the neighbours effectively reduce from 4 (or 6 in the next levels) to 1. The same concept extends to further neighbours. The neighbours are grouped by their distance from the original vertices. Typically we require a new parameter per each group of neighbours. With this symmetry we reduce the number of parameters along the columns of $\mathbf{A}$ (and rows of $\mathbf{P}$). (The reduction factor in this case depends on the definition of the neighbour groups.) With this method we obtain a neighbour structure for each matrix, we will call them $\mathbf{A}_{\text{stru}}$ and $\mathbf{P}_{\text{stru}}$.

This approach can be obviously made recursive along the different levels of the subdivision.

Introducing these symmetries we reduce considerably the number of independent parameters in the problem. The actual matrices $\mathbf{A}$ and $\mathbf{P}$ can be reduced to a list of degrees of freedom to be fed to the minimization algorithm. We can design two functions: one that reduces the two matrices to a vector (downscale) and one that recovers the full matrices from the unknowns vector (upscale).

As a consequence of this reduction of degrees of freedom, we face two challenges:

- Implementation challenge: the minimization algorithm sees only the independent parameters that now are reduced in number, but to calculate the cost function needs the full matrices are needed. The jacobian and hessian of the cost function have to be calculated only with respect to the independent parameters. How does this fit into the automatic differentiation implementation?

- The number of constraints reduces in a non-trivial manner: the constraints involve a matrix product of the full matrices. We have to figure

out which constraints are effectively independent after the reduction of degrees of freedom.

In the Sections B.2 and B.3 we will illustrate our approach to these problems.

## B.2 Implementation of DOF Reduction Techniques Inside the Automatic Differentiation

In this Section we will show how to calculate the automatic derivatives of a function, with respect to its 'true variables'. In the following code example, we define our vector of unknowns as **x**, that has dimension 3. The `function` mimics the cost function of our minimization problem, in a much more simple fashion. The first operation performed inside the `function` is to grow the vector **x** into two matrices, `upscaled_A` and `upscaled_B`. Then the two matrices are multiplied together (@) and an identity matrix (`torch.eye`) is subtracted. The result of these operations is summed up to obtain a scalar, as for any typical cost function. The `function` returns the value of the cost function and **x**, that somehow, in disguise, is gone through all the operations described. We then can take the derivative of the function and see if PyTorch is able to correctly calculate the derivatives of this function with respect to **x**.

```
1  # upscaling of variables
2  # and calculate derivatives of a function incorporating
   ↪  up/downscaling
3
4  import torch
5  import hessian as h
6
7
8  # define two variables
9  x = torch.tensor([1.5, 2.5, -5.5], requires_grad=True,
   ↪  dtype=torch.float32)
10
11
12 def function(x):
```

```
    # upscale tensor
    rows = 2
    cols = 2
    upscaled_A = torch.zeros(rows, cols, requires_grad=True,
     ↪  dtype=torch.float32)
    upscaled_B = torch.zeros(rows, cols, requires_grad=True,
     ↪  dtype=torch.float32)

    upscaled_A[0] = x[[0, 1]]
    upscaled_A[1] = x[[1, 0]]
    print("Matrix A:")
    print(upscaled_A)

    upscaled_B[0] = x[[2, 1]]
    upscaled_B[1] = x[[1, 2]]
    print("Matrix B:")
    print(upscaled_B)

    cost = upscaled_A @ upscaled_B - torch.eye(rows)
    cost = torch.sum(cost)
    return cost, x


f_torch, x = function(x)
print("Function value:")
print(f_torch.data)

# calculate its jacobian
jac = h.jacobian(f_torch, [x])
# print result
print("Jacobian")
print(jac)
```

---

Gives this output:

```
Matrix A:
tensor([[1.5000, 2.5000],
        [2.5000, 1.5000]], grad_fn=<CopySlices>)

Matrix B:
```

```
tensor([[-5.5000,  2.5000],
        [ 2.5000, -5.5000]], grad_fn=<CopySlices>)

Function value:
tensor(-26.)

Jacobian
tensor([[-6.,  2.,  8.]])
```

It is important to note that the Jacobian has the correct dimension, having $\mathbf{x}$ dimension 3, the Jacobian will be of dimension 3 as well:

$$Jf(\mathbf{x}) = (\partial f/\partial x_1, \partial f/\partial x_2, \partial f/\partial x_3)$$

and it is exactly what we get.

This example is quite simple, and it might look trivial, but it is not. The non-trivial part is the "upscaling", where $\mathbf{x}$ is grown into two different matrices. PyTorch is able to propagate the derivatives through this operation, which is not one of the elementary operations we mentioned in Appendix A.

The same method works for the derivatives of the constraints, that are defined in a very similar way, e.g. $\mathbf{AP} = \mathbf{1}$. The challenge with the constraints is to identify the independent constraints, now that the number of DOFs has been reduced dramatically, i.e. not all the equations produced by $\mathbf{AP} = \mathbf{1}$ are linearly independent.

## B.3 Method for Reduction of Constraints

From the reduction of variables (just described), we obtain a neighbour structure for each matrix. With this structure we can calculate the constraints and then isolate the independent ones.

As an example, as before we will take the first stage of the minimization. The constraints for this stage are the ones defined by $\mathbf{AP} - \mathbf{1} = \mathbf{0}$, as already mentioned. Having this neighbour structure, we rewrite the constraints as $\mathbf{A}_{\text{stru}}\mathbf{P}_{\text{stru}} - \mathbf{1} = \mathbf{0}$. This equation defines a set of non-independent equations. We want to figure out which are the independent equations, that are our remaining constraints after the reduction of variables. Operatively, we put this

linear system of equations in matrix representation and use the reduced row-echelon form [Beezer, 2012] to identify the independent constraints.