Speech Analysis with Production Constraints

Sacha Krstulović

Ph.D. thesis submitted to the Department of Electricity of the EPFL
Lausanne, Switzerland

Electronic copies of this document are available as :

ftp://ftp.idiap.ch/pub/sacha/Thesis/thesis.{ps,ps.gz,pdf}

or through the author's web page :

http://www.idiap.ch/~sacha/

Digest

State of the art speech analysis and feature extraction techniques rely mainly on simplified auditory models (e.g. the Mel scale or PLP features). These systems can model any sound and are not particularly specialized in the modeling of speech. Hence, they fail to reflect some typical speech characteristics such as co-articulation.

To bridge this gap, we propose to match some speech production paradigms with Automatic Speech Processing technologies (ASP). This matching is based *(1)* on an analogy between the Linear Prediction (LP) of speech and the acoustic modeling of lossless tubes, and *(2)* on the fact that most of today's state of the art speech production models use an acoustic tube as the interface between the acoustic level and the speech production strategy level. In this framework, we develop two innovative feature extraction methods : the Non-Uniform Topology (NUT) analysis, and the "Relating Acoustics to a Linear Shape Model" (ReALiSM) method.

To establish the **NUT analysis**, we begin with generalizing the traditional LP lattice filter/lossless tube equivalence to the case of tubes discretized in unequal-length sections, such as the non-uniform tube used for the Distinctive Regions and Modes (DRM) speech production model.

We show that imposing unequal-lengths to the tube sections is equivalent to constraining some reflection coefficients to stay zero-valued in the corresponding lattice filter. This "Non Uniform Topology" constraint allows to de-couple the number of degrees of freedom (DoFs) of the model from the dimensions of its acoustic counterpart (given by the number of poles). To use this new model as an analysis tool, we derive some relevant parametric estimators, based on the analytic minimization of a well-defined error criterion. Finally, remarking that a fixed non-uniform topology (e.g., the one of the DRM production model) may not be optimal for every part of speech, we propose a method to optimize the repartition of the lengths/delays.

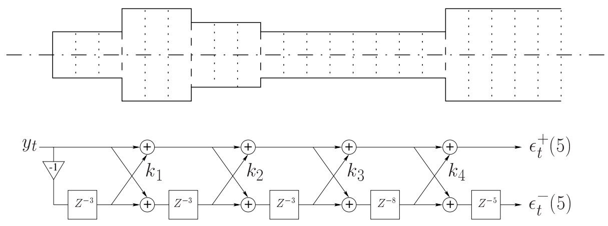
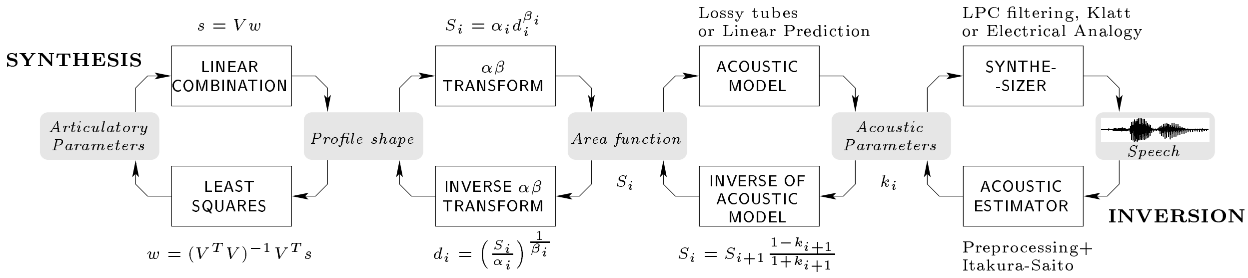


Fig.1 : a NUT tube and the equivalent lattice filter.

The assessment phase shows that NUT models permit a significant reduction in the number of parameters necessary to describe a speech spectrum, while keeping a high level of spectral accuracy. It is verified that they consistently produce a lower residual error than the unconstrained filters with and equal number of DoFs. It is also verified that the NUT models are consistent with spectral analysis since the topology optimization helps minimizing the spectral distortion induced by the reduction of the number of DoFs. Moreover, the tube topologies themselves may be used as an analysis tool.

Alternately, to establish the **ReALiSM method**, we implement a projection of the solution of inverse LP lattice filtering into the parameter space of Maeda's linear vocal tract shape model, through a series of linear and non-linear transformations :



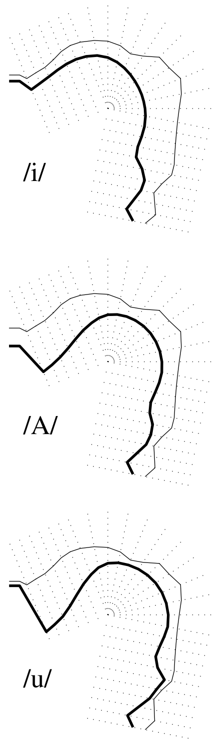


Fig.2 : Inversion of human vowels.

The assessment of this system is realized in two phases. First, it is verified that information losses that occur within Least-Squares smoothing, area functions resampling and reflection coefficients estimation still allow for recovery of synthetic template tract shapes. A set of synthetic vowels is produced (cardinal French vowels registered in the UPSID phonemic database), and informal listening tests ensure that they are acceptable despite the fixed length approximation and the lossless LPC synthesizer. Subsequent inversion results show that the estimated shapes are close to the original synthetic shapes.

In a second phase, the system is used to invert real speech recorded from a French male speaker in a quiet environment. Several vowel sequences and VCV sequences are tested. For example, results corresponding to the vowels /i A u/ are given on figure 2. The system locates cavities at phonetically relevant places of articulation (e.g. front for /A/, back for /i/). Lip apertures are also realistic, e.g., in an /A b i/ sequence the /b/ consonantal closure is detected. This system offers significant advantages over existing acoustico-articulatory inversion schemes, such as real-time computation, modularity and links with Digital Signal Processing techniques.

Since Linear Prediction analysis is used as a feature extraction method in most of the main ASP technologies (such as speech coding, speech/speaker recognition, speech synthesis, speech enhancement etc.), the production-based analysis methods that we have developed create a new gateway for the **integration of speech production constraints in the main classical ASP applications**. To assess the benefits that these methods introduce, we propose multiple ways to exploit the NUT and ReALiSM systems in various branches of the ASP domain. In particular, we provide and discuss encouraging preliminary speech recognition results.

Version abrégée

Les techniques de l'état de l'art en analyse de la parole et extraction de paramètres caractéristiques reposent principalement sur des modèles simplifiés de l'appareil auditif (par ex. avec l'échelle Mel ou les paramètres PLP). Ces sytèmes peuvent modéliser n'importe quel son, et ne sont pas particulièrement spécialisés à la modélisation de la parole. Ainsi, ils échouent à refléter des caractéristiques spécifiques à la parole, telles que la co-articulation.

Pour combler cette lacune, nous proposons d'intégrer certains paradigmes de production de la parole aux technologies du Traitement Automatique de la Parole (TAP). Cette intégration est basée *(1)* sur une analogie entre la Prédiction Linéaire (LP) et les modèles acoustiques de tubes sans pertes, et *(2)* sur le fait que plusieurs des modèles de l'état de l'art en production de parole emploient un tube comme interface entre le niveau acoustique et celui de la stratégie de production. Dans ce cadre, nous développons deux méthodes innovantes pour l'extraction de paramètres caractéristiques : l'analyse à topologie non-uniforme (NUT en anglais), et une méthode reliant l'acoustique à un modèle linéaire de forme de conduit vocal (ReALiSM en anglais).

Pour établir l'**analyse NUT**, nous commençons par généraliser l'équivalence traditionnellement établie entre la prédiction linéaire et les modèles de tubes sans pertes au cas où les tubes sont discrétisés en sections de longueurs inégales, comme dans le cas du modèle de production à régions distinctives (DRM).

Nous montrons que l'imposition de sections inégales est équivalente à contraindre certains des coefficients de réflection du filtre associé à garder une valeur nulle. Cette contrainte de "Topologie Non-Uniforme" permet de dé-coupler le nombre de degrés de liberté (DdLs) du modèle de la dimension de sa contrepartie acoustique (donnée par le nombre de pôles). Pour utiliser ce nouveau modèle en analyse, nous dérivons des estimateurs paramétriques adéquats, basés sur la minimisation analytique d'un critère d'erreur bien défini. Enfin, en remarquant qu'une topologie non uniforme fixe (telle que celle du modèle DRM) peut ne pas être optimale pour tous les constituants de la parole, nous proposons une méthode pour optimiser la répartition des longueurs/des retards de filtrage.

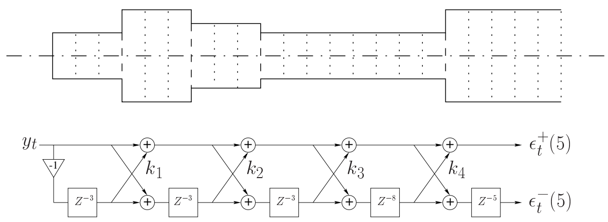
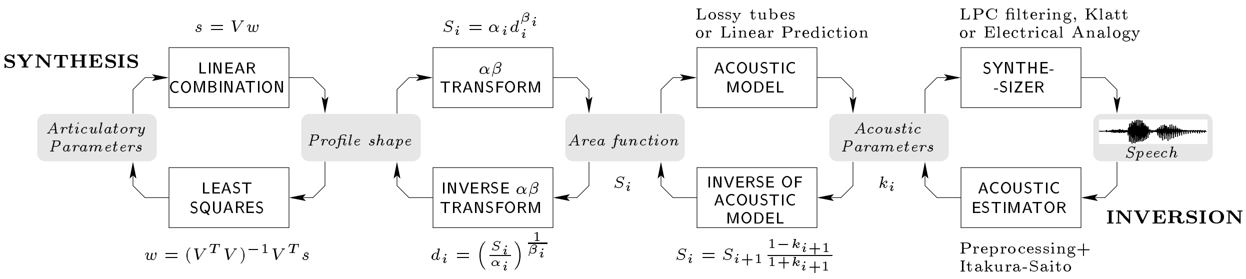


Fig.1 : Un tube NUT et le filtre en treillis équivalent.

La phase de validation montre que les modèles NUT permettent de réduire significativement le nombre de paramètres nécessaires à la description d'un spectre de parole, tout en gardant un haut degré de précision. Il est vérifié qu'ils produisent de manière consistante une erreur résiduelle moindre que les filtres non contraints à nombre de DdLs égal. Il est aussi vérifié que les modèles NUT sont en accord avec l'analyse spectrale grâce à l'optimisation de la topologie, qui permet de minimiser la distortion spectrale relative à la réduction du nombre de DdLs. En outre, les topologies peuvent en elles-même être utilisées comme un outil d'analyse.

Par ailleurs, pour établir la **méthode ReALiSM**, nous réalisons une projection de la solution du filtrage linéaire inverse vers l'espace des paramètres du modèle de conduit vocal de Maeda, à travers une séries de transformations linéaires et non-linéaires :



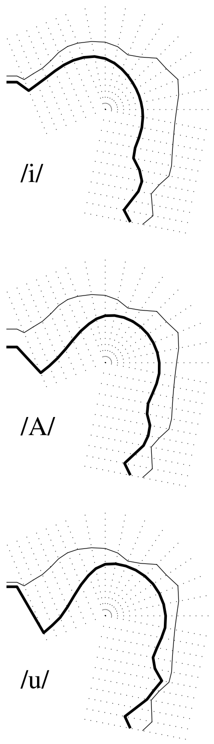


Fig.2 : Inversion de voyelles humaines.

La validation de ce système est réalisée en deux phases. Premièrement, il est vérifié que les pertes d'information relatives au lissage par moindres carrés, au ré-échantillonnage de la fonction d'aire et à l'estimation des coefficients de réflection permettent de retrouver des formes de conduit vocaux synthétiques imposées. Pour cela, un ensemble de voyelles synthétiques est produit (voyelles cardinales françaises de la base UPSID), et des tests informels d'écoute assurent qu'elles sont acceptables malgré l'approximation à longueur fixe et malgré le synthétiseur LPC sans pertes. Les résultats consécutifs montrent que l'inversion de ces voyelles produit des formes proches des formes synthétiques originales.

Dans une seconde phase, le système est utilisé pour inverser de la parole réelle enregistrée dans un environnement silencieux par un locuteur français. Plusieurs séquences de voyelles et de VCV sont testées. Par exemple, les résultats correspondants aux voyelles /i A u/ sont donnés dans la figure 2. Le système localise les cavités au niveau de lieux d'articulation raisonnables d'un point de vue phonétique (par ex., en avant pour le /A/, en arrière pour le /i/). Les ouvertures aux lèvres sont également réalistes, par ex., dans une séquence /A b i/, la fermeture consonnantique du /b/ est détectée. Ce système offre des avantages significatifs par rapport aux systèmes d'inversion acoustico-articulatoire existants, comme le calcul en temps réel, la modularité et des liens avec les techniques du Traitement Numérique du Signal.

Comme l'analyse par Prédiction Linéaire est à la base de l'état de l'art des techniques d'extraction de paramètres utilisées par la plupart technologies du TAP (telles que le codage, la reconnaissance de parole ou de locuteur, la synthèse, le débruitage etc.), les méthodes d'analyse que nous proposons, basées sur la production, créent une nouvelle passerelle pour l'**intégration de contraintes de production dans les principales applications du TAP**. Pour évaluer les bénéfices que nos nouvelles méthodes introduisent, nous proposons plusieurs manières d'exploiter les systèmes NUT et ReALiSM dans diverses branches du TAP. En particulier, nous fournissons et discutons des résultats préliminaires encourageants en reconnaissance de la parole.

To my family:
Gisèle, Nikša, Sylvie and Serge,
and all those who showed pride
for this thesis.

To Mehdi Chétioui,
my best friend in Switzerland,
for his hearty and sincere
moral support.

Acknowledgements

This work has been supported by the Swiss National Science Foundation, under grants Nr. 2100-49.725.96 and Nr. 20-55.634.98 for the ARTIST project.

The Institute Dalle-Molle for Perceptual Artifical Intelligence (IDIAP) has been my hosting institution and has supported part of the financing of this work.

The Social Service of EPFL has helped me to survive during the last two months.

Thanks to you all !

Thanks to the members of the PhD exam Jury for having accepted the charge of judging this work:
Dr. Frédéric Bimbot, Prof. Hervé Bourlard, Prof. Martin Hasler, Dr. Gernot Kubin, Prof. Philippe Robert.

A student doesn't go without a teacher. The following colleagues helped me greatly to progress through scientific and/or moral support:
Frédéric Bimbot, René Carré, Gérard Chollet, Martin Hasler, Yves Laprie, Shinji Maeda, Ivan Magrin-Chagnolleau, Chafic Mokbel, Pascal Perrier.

Thanks to Dr. J.H. Westbury, Dr. C. Johnson and the Waisman Center of the University of Wisconsin for their free ditribution of the UW-XRMB database.

I have had particularly fruitful and/or friendly exchanges with the following former or present IDIAP members:
Astrid, Alessandro, Beat, Cédric, Christopher, Dominique, Eddy, Frédéric, Frank, Georg, Gilbert, Gilles, Hans, Hubert, Jean-Luc, Johnny, Jürgen, Katrin, Kim, Miguel, Nicolas, Olivier, Perry, Rachel, Ronan, Samy, Souheil, Sylvie, Thierry, Todd.

The following people are my friends: they encouraged me, gave me a smile or lent me an ear, spared with me or poured me an apple juice in the hard times. Big respect to:
Annick and Trevor, the Barock Team (Carole, Mister Kim, Nath), the Brit'Team (Anjelica, Anna, Christian, Jess), Capoeira ABADÁ Valais (Gr. Lobaõ and every Roda player), Claire, Didier, DRR, Edite, Floriane, Fred and Véro, Germinal, Jacqueline, Katiana, the Lord Sandwich Team (Anna, Chimène, Inès, Véronique and Véronique), the Loup Blanc Team (Angélique, Dalia, Lionel, Michel, Hélène, Tila), Malika, Martine, Mehdi, the Migros Catering Team, Rintze and Ingrid, Samuel, Sei Mei Kan Aïkido Dojo of Martigny (Werner and everyone who shared some 気 with me), Sensei Ikeda, Tao and his mute cousin, DjerryC, Véro and J.B.

"For unity can keep humans brave." (Khaïlé Sélassié)

Contents

List of Tables

List of Figures

List of Abbreviations

AAI	Acoustico-Articulatory Inversion
ASP	Automatic Speech Processing
ASR	Automatic Speech Recognition
DoF	Degree of Freedom
DRM	Distinctive Regions and Modes
	(the speech production model proposed by Mrayati et al. [1988])
DSP	Digital Signal Processing
CELP	Code Excited Linear Prediction
HMM	Hidden Markov Model
LAR	Log Area Ratio
LPC	Linear Prediction Coding
LPCC	Linear Prediction Cepstral Coefficient
MFCC	Mel Frequency Cepstral Coefficient
MSE	Mean Squared Error
NUT	Non-Uniform Topology
ReALiSM	"Relating Acoustics to a Linear Shape Model"

Chapter 1

Introduction

One of the beauties and curses of the Automatic Speech Processing techniques (ASP) is that they aim at interfacing the rigid and deterministic world of computers with the purely human, randomness-prone phenomenon of speech. While computers always give the same output when processing the same data, the acoustic realization of a particular suite of phonemes can vary across a noticeable scale without breaking the underlying message. The variability of speech can be of two kinds:

- extrinsic variability, the one introduced by the environment where the speech signal is collected;
- intrinsic variability, which arises from coarticulation phenomena, physiological differences between speakers, dialogue context or emotional state of the speaker.

Dealing with these sources of variability is one of the main challenges that most ASP applications have to face. As a matter of fact, these variabilities constitute sources of information that are spurious to the pure phonetic content of speech, and that one does not necessarily want to recognize (in the framework of speech recognition) or to transmit (in the framework of speech coding). Conversely, one may wish to exploit these variabilities independently of the phonetic content, e.g. in the framework of speaker recognition, speech synthesis or auditory scene analysis.

In today's state of the art recognition and coding systems, the extrinsic variability is mainly dealt with at the feature extraction level, through the use of additive and convolutional noise cancellation techniques (such as spectral subtraction and blind deconvolution, e.g. cepstral mean subtraction [Lockwood and Boudy, 1991; Hermansky and Morgan, 1994; Mokbel et al., 1997]). But so far, less attention has been paid to the definition of features that would allow to isolate the sources of intrinsic variability of speech.

It is usually stated that articulatory-based features would be suitable candidates for a better modeling of the intrinsic variability. As a matter of fact, they would allow a characterization of speech in relation to its production process. The production process is considered more stable (less intrinsically variable) than its acoustic counterpart. The review of the arguments supporting these assertions will be the occasion to expose the general theoretic framework of the present thesis (section 1.1).

The above statements have found support in a domain of human sciences known as Articulatory phonology, which will be introduced in section 1.1.1. This subject has boosted the study of a class of problems known as Acoustico-Articulatory Inversion (AAI), aiming at recovering articulatory gestures from the speech sound alone, and described in section 1.1.2. But we will see that despite the many existing AAI methods, the exploitation of articulatory knowledge in Automatic Speech Processing remains a challenge (section 1.1.3).

Rather than trying to recover accurate gestures from the sound of speech, we propose to constrain the estimation of LPC features with respect to speech production constraints (section 1.2). This idea is viewed as a means of allowing articulatory knowledge to penetrate the analysis schemes used in state of the art ASP applications. The rationale leading to this idea is presented in section 1.2.1, while the practical realization of it is introduced in section 1.2.2.

After the exposition of the general theoretic framework and of the thesis position, an overview of this report will be given in section 1.3.

1.1 General theoretic framework

1.1.1 Articulatory phonology and the motor theory of speech perception

Starting from a cognitive point of view, the Haskins Laboratories have developed the Motor Theory of Speech Perception [Liberman et al., 1967; Liberman and Mattingly, 1985], which states that a listener uses articulatory a-priori knowledge, related to his/her ability to produce speech gestures, in the course of speech or speaker recognition. This implies that human beings use articulatory clues in the process of recognizing speech.

Browman and Goldstein [1987, 1990] have deepened this idea by establishing the Articulatory Phonology theory. This theory states that speech is the product of overlapping dynamical regimes, called gestures, which regulate the shape of the vocal tract. From that perspective, the actual units of speech are gestures. Variability in the synchrony of gestural units during the process of acoustic production of speech gives birth to the intrinsic variability known as coarticulation.

Exploiting these conclusions into current speech recognition or modeling systems would mean performing recognition or coding of more stable gestural/articulatory patterns rather than acoustic features. As a consequence, recognition or coding performance may improve.

We will see in chapter 5 that the expected improvement has actually been observed in several speech recognition experiments. But as a prerequisite to the application of articulatory paradigms in recognition or coding, we will first see that the use of acoustic-to-articulatory inversion is necessary to have access to some articulatory features.

1.1.2 Acoustico-articulatory inversion

While experimental articulatory speech processing systems can make use of recorded articulatory data, real-life systems have to use acoustico-articulatory inversion to have access to some articulatory features. As a matter of fact, devices able to record articulatory data (X-ray devices [Bothorel et al., 1986; Westbury, 1994], scanners [Story et al., 1996] or electro-magnetic articulographs [Cartsens and Carstens, 2001]) are expensive and invasive, and most of all, incompatible with some potential everyday life applications. Hence, it is desirable to extract the articulatory information from the speech waveform. In this case, the microphone stays the main speech input device, and a reliable acoustic-to-articulatory inversion scheme has to be developed[1].

The acoustic-to-articulatory mapping is a "one-to-many" inversion problem that has no direct analytical solution. Indeed, given a vocal tract configuration and a defined excitation signal, the speech signal at the output of the vocal tract is unique. But conversely, several vocal tract configurations can produce the same acoustic signal [Atal et al., 1978; Bonder, 1983]. In mathematical terms, the transformation between the articulatory space and the acoustic space is *not* an homeomorphism : it is a surjection from the articulatory space to the acoustic space. This problem has been largely studied in the past [Schroeder, 1967; Mermelstein, 1967; Atal et al., 1978; Schroeter and Sondhi, 1994], and several approaches have attempted to overcome or regularize the ill-posed nature of the problem. They can be classified into :

- **Codebook approaches** [Atal et al., 1978; Shirai and Honda, 1980; Charpentier, 1984; Schroeter et al., 1990; Yu, 1993; Ouni and Laprie, 2000], where the articulatory space is quantized and the corresponding acoustic features are synthesized to form a codebook of acoustic/articulatory vector pairs. The inversion is implemented as a search in the codebook, and coarticulation is modeled as constraints ruling the possible search paths.
- **Neural network approaches** [Atal and Rioul, 1989; Rahim and Goodyear, 1990; Soquet et al., 1990; Laboissière et al., 1991; Shirai and Kobayashi, 1991; Rahim et al., 1993], where the parameters of some neural networks are trained to get a nonlinear continuous mapping

[1] Video-based inputs are also more and more used, allowing to perform "audio-visual" speech recognition [Yehia et al., 1997; Luettin, 1999; Dupont and Luettin, 2000]. Methods for segmenting face pictures sometimes specifically aim at extracting articulatory features such as the lip shape or the jaw opening, with the goal of using these features in the recognition system. Aspects related to such video-based methods won't be dealt with in the present thesis.

between the articulatory parameters and the acoustic features. In this case, coarticulation is modeled through topological constraints imposed to the network.

- **Constrained optimization approaches** [Flanagan et al., 1980; Levinson and Schmidt, 1983; Shirai and Kobayashi, 1991; Prado et al., 1992], where classical iterative optimization techniques are used to drive the parameters of an articulatory synthesis model in order to minimize a distance between a template acoustic feature vector and a synthesized one. Various constraints (continuous mapping functions, speed of articulatory movements, ...) can be introduced in order to overcome the ill-posed nature of the inversion problem.

- **Analytic approaches**, where the correspondence between the acoustic features and the articulatory parameters is explicitly defined by some analytic expressions. The analytic approaches in themselves can be based on different methods:

  - the *variational method* [Jospa et al., 1994; Ciocea and Schoentgen, 1998; Laprie and Mathieu, 1998], where variational calculus is used to match acoustic trajectories with articulatory trajectories. This method includes inherent coarticulation constraints in the definition of an energy function to be minimized analytically;
  - the *Fourier expansion based approach* [Mermelstein, 1967; Schroeder, 1967; Mokhtari, 1998; Mokhtari and Clermont, 2000], where the parameters of a Fourier expansion of the vocal tract shape are shown to correspond, under certain conditions, with the parameters of the speech formants;
  - the *Linear-Prediction coding (LPC) based approach*, where the propagation of sound waves through acoustic tubes (in our case, vocal tracts) is modeled by Auto-Regressive (AR) modeling techniques [Wakita, 1972; Wakita and Gray Jr., 1975; Wakita, 1979; Scaife, 1989]. In this method, well-known Digital Signal Processing estimation methods are used to recover articulatory parameters. This technique facilitates the inversion problem because it has a clear mathematical formulation, but it does not overcome completely the ill-posed nature of the problem because it produces sensitive solutions. As for today, it doesn't incorporate articulatory constraints that would allow to produce more robust solutions.

- **Stochastic modeling and statistical inference methods**, which bring an attractive solution to inverse problems since they allow to efficiently model many-to-one mappings [Ghahramani, 1993]. As a matter of fact, the whole set of solutions to an inversion problem can be represented by some mixtures of conditional probability density functions. Furthermore, some coarticulation constraints can be learned from the data (by performing conditional probability measures) rather than being imposed from some a-priori phonetic knowledge. The vogue for this class of solutions has only started recently. Different methods have been proposed, making use of:

  - *Mixture Density Networks* [Richmond, 2001], which provide a multi-modal stochastic mapping after training with a non-linear optimization algorithm;

– *mixtures of dynamical systems* (Kalman filters) [Ramsay, 1996; King and Wrench, 1999; Frankel and King, 2001], where speech is modeled as a random walk in a graph connecting several dynamical regimes. Each separate dynamical system corresponds to the articulatory realization of a particular phoneme. The training of this system is performed via the EM algorithm;

– *Hidden Markov Models (HMMs) with an articulatory state space* [Deng and Sun, 1994; Erler and Freeman, 1996; Richardson et al., 2000a], where coarticulation is implicitly modeled by the topology imposed to the state transitions graph;

– *Maximum Likelihood Continuity Mapping* [Hogden, 1996], which broadens the articulatory HMMs idea by proposing to infer the model topology in a non-supervised way. This is made possible by imposing some constraints which rule the continuity and the speed of motion from one articulatory state to the other. Based on these constraints, the state space is automatically structured by a Maximum Likelihood method rather than manually structured through a supervised design of the model's transition graph;

– *Bayesian Networks* [Stephenson et al., 2000], where some probabilistic articulatory dependencies are explicitly included as hidden states in the transition graph, rather than implicitly modeled in the network topology.

The four later methods are specifically designed for speech recognition applications: in those cases, acoustico-articulatory inversion is a byproduct of the decoding phase and does not suit any particular speech analysis interest.

1.1.3 The problem of using speech production knowledge in ASP

Apart from the stochastic methods, most of the exposed approaches give practical solutions to the inversion problem but do not provide an accurate knowledge of the acoustico-articulatory mapping topology that would go beyond the injective ("one-to-many") nature of the mapping. This is a severe problem, because without more knowledge it is impossible to relate statistics in the acoustic domain to statistics in the articulatory domain. In particular, it is yet impossible to provide articulatory statistics that would relate to an acoustic distance. Alternately, the definition of a purely articulatory distance is not available[2]. These problems have prevented a successful application of the evoked methods in the frameworks of speech coding and speech/speaker recognition.

Alternately, the stochastic approaches may solve the problem, but they require a sufficient amount of data to train the models. As a matter of fact, the acoustico-articulatory databases which have been available so far, such as the Strasbourg X-ray Cineradiographic database [Both-

[2]The definition of a distance is necessary both in speech/speaker recognition, to match an observation against a set of models, and in speech coding, to allow for an efficient quantization of the transmitted features.

orel et al., 1986], the University of Wisconsin X-ray Micro-Beam database (UW-XRMB) [Westbury, 1994] and the ATR X-ray film database [Munhall et al., 1995], have been designed for the needs of phonetic studies: they usually don't provide enough data for an accurate training of a mapping model. Furthermore, they often require an extensive preliminary work of resource management, including image or feature segmentation, hand-made or automatic labeling, and listening assessment. The recent boost of the stochastic mapping approaches is indeed linked with the appearance of more adequate databases, such as the MOCHA acoustico-articulatory recordings [Wrench and Hardcastle, 2000].

From that point of view, the exploitation of speech production knowledge in the ASP framework appears to be a still open research issue.

1.2 Position of the thesis

1.2.1 Our manifesto

One workaround to the evoked problems is to try and constrain the extraction of acoustic features on the basis of articulatory considerations, rather than trying to extract pure articulatory features.

A similar reasoning has proven fruitful when including auditory paradigms in speech analysis. The contemporary state of the art speech recognition/coding systems have indeed broadly benefited from auditory notions such as the Mel scale [Rabiner and Juang, 1993], critical band spectra and loudness curves [Hermansky, 1990; Hermansky and Morgan, 1994] or more elaborate perceptual models [Painter and Spanias, 2000]. In the corresponding approaches, the purpose was not to include the precise quantitative characteristics of somebody's ear in the extraction process, but rather to exploit some of the general human ear principles with the enhancement of a particular application in mind. As a matter of fact, the Mel scale used for the Mel Frequency Cepstral Coefficients (MFCCs) or the loudness curves used in RASTA analysis correspond only roughly to human characteristics. Nevertheless, they help reducing bit-rates or enhancing speech recognition performances.

In regard, few concluding proposals have been made concerning the use of speech production paradigms for speech analysis. The Vocal Tract Length Normalization (VTLN) techniques [Zhan and Waibel, 1997] seem to be the only relevant example.

Hence, the position we want to adopt for the present thesis is the following. Speech Production knowledge could be used to reduce the number of degrees of freedom in the feature extraction process, thus allowing for less intrinsic variability. We would like this speech production knowledge to be directly represented into the speech *features* (or into the speech *code*), rather than implicitly represented in the structure of a wider acoustico-articulatory model. Independently of their degree of anthropomorphic accuracy, these features may help improving some end ASP applications.

The feature extraction systems most commonly used in nowadays' state of the art ASP systems are based either on non-parametric spectral analysis techniques, with "FFT-based" parameters such as filterbank outputs or MFCCs, or on parametric spectral analysis techniques, with Linear Prediction Coding techniques such as the Levinson recursion [Rabiner and Juang, 1993] applied to the extraction of reflection, prediction or cepstral coefficients. Alternately, it is well known that the LPC techniques have been designed in relation to crude vocal tract models [Markel and Gray, 1976; Wakita, 1979]. We have also seen that the LPC-based acoustico-articulatory inversion methods provide a clear mathematical formulation of the acoustico-articulatory mapping. From that perspective, building upon the LPC-based acoustico-articulatory inversion approaches and matching them with state of the art parametric feature extraction techniques appears to be the best way to achieve our goal.

Throughout the present thesis, we will therefore stick to the theoretic framework of LPC modeling through inverse filtering and use this framework as the gateway for the introduction of more elaborate speech production constraints in state of the art speech analysis methods.

1.2.2 Practical realization of our objectives

Now, the question arises as how to include more speech production knowledge in the LPC modeling framework. The answer we propose [Krstulović, 2000a] stems from the following remark : most of the speech production models known today use an area function as the interface between the articulatory level and the acoustic modeling level (figure 1.1).

The area function describes the repartition of the cross-sectional areas of the modeled vocal tract as a function of the position along the curvilinear tract axis. The prior role of the speech production model is to constrain the shapes to correspond to some realistic speech production characteristics, while the subsequent role of the acoustic model is to describe the filtering effects of the shaped tubes in terms of acoustic features.

As a matter of fact, the LPC-based AAI methods allow to relate the area function to some particular acoustic parameters deriving from the LPC analysis, and known as the reflection coefficients. This relation uses a simple non-linear expression. By reflecting the shape constraints arising from a speech production model in the LPC analysis process via the area function and the reflection coefficients, our objective will be met.

Among the available speech production models, two classes of models, corresponding to different production constraints, will be employed to put the proposed idea in practice :

- the models considering that the vocal tract is a series of tubes of unequal length, such as the Distinctive Regions Model (DRM) [Mrayati et al., 1988; Krstulović, 1996];

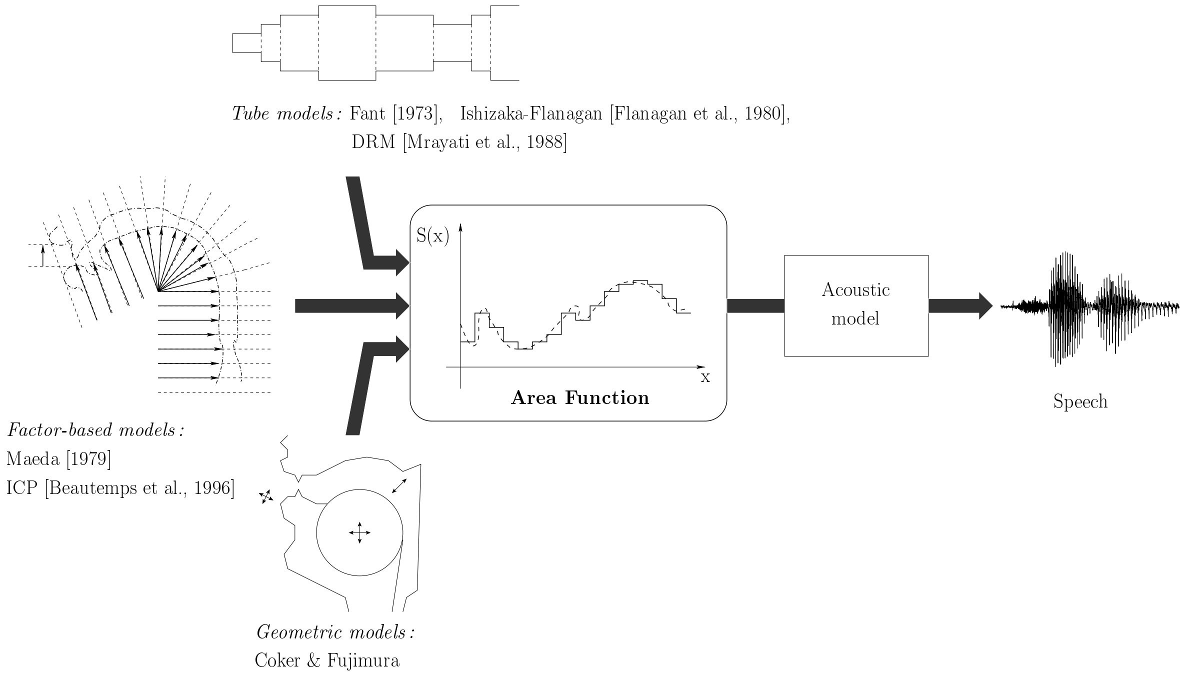


Figure 1.1: From speech production models to speech acoustics via the area function.

- the models considering that the shape of the sagittal cut of the vocal tract can be modeled as a linear combination of articulatory factors, such as in Maeda's model [Maeda, 1979] or in the related ICP model [Beautemps et al., 1996].

More precise definitions of these models, plus a more detailed explanation of the relationship between the area functions, the reflection coefficients and the other LPC coefficients will be given in the relevant chapters of this report.

1.3 Overview of the thesis

This introductory chapter described the theoretical framework and the position of our work. It stated our objective, which consists in incorporating speech production constraints in the state of the art speech analysis methods via the exploitation of a possible link between LPC modeling and some well known speech production models.

Chapter 2 will actually expose the relationship between LPC analysis and vocal tracts seen as lossless tubes. In this part, and for the need of our study, we will extend the traditional analogy to the case where the tubes are made of a concatenation of unequal-length sections.

In chapter 3, we will expose how these non-uniform tube models (NUTs) can be used as a speech analysis tool. Experimental results will show that they allow to achieve a lower modeling error than traditional LPC models with the same number of degrees of freedom. Hence, they allow for a dimensionality reduction of the feature space, while retaining a good spectral modeling accuracy.

Chapter 4 will expose the developments related to the exploitation of a factor-based production model. Results will show that our LPC-based shape extraction method is able to capture some of the human speech production characteristics, while being based on a fast analytic method.

Chapter 5 will expose the benefits that the developed feature extraction methods can bring to some selected ASP applications.

We will conclude by summarizing the contributions of the thesis.

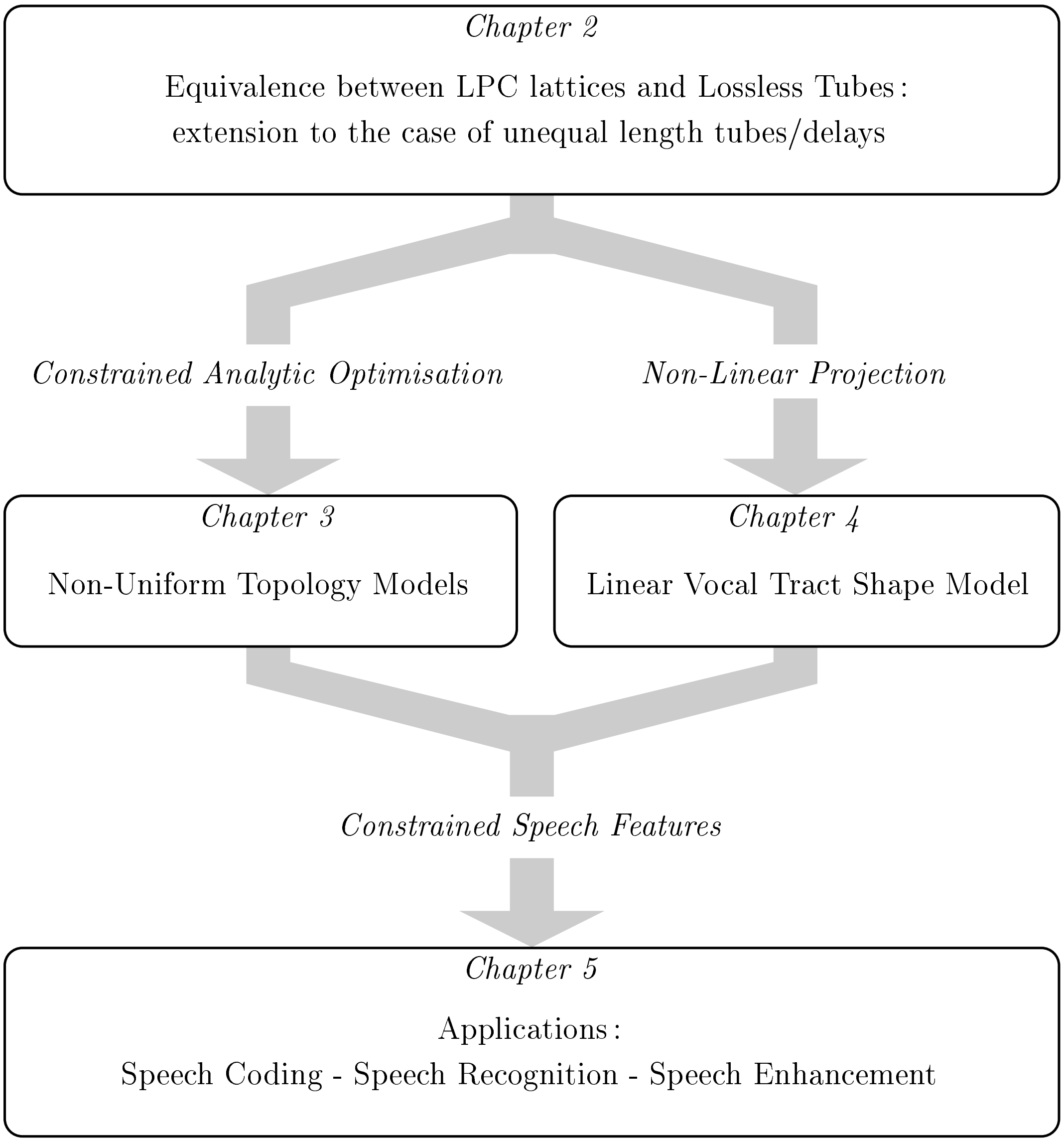


Figure 1.2: Organization of the thesis.

Chapter 2

Relating LPC to articulation: the lossless tube model

It is traditionally considered that the LPC parameters extracted from the speech signal account for the changes in the vocal tract configuration, while the corresponding residual signal plays the role of the excitation produced by the vocal cords. Authors such as Wakita [1972, 1979], Markel and Gray [1976] and Bonder [1983] have actually established a formal analogy between LPC and the process of wave propagation into lossless discrete acoustic tube models, possibly used as vocal tract models.

As such, LPC constitutes a crude speech production model. Indeed, the formal analogy classically implies that the individual portions forming a discretized tube all have a unit length. Equivalently, on the Digital Signal Processing (DSP) side, the model does not incorporate any other a-priori knowledge about the signal being modeled than the all-pole nature of the corresponding filter. On both sides, the model is unspecialized: it could be applied to any signal, and is not bound to model additional a priori knowledge about speech production in particular.

On the other hand, some speech production models issued from the phonetic sciences, such as the Distinctive Regions Model (DRM, [Mrayati et al., 1988; Krstulović, 1996]), consider that tubes with unequal-length sections are better suited to a realistic driving of the formant trajectories. Alternately, taking up the LPC/tube analogy, there is no reason for the tube representing the vocal tract to have as many degrees of freedom as the resulting spectral model.

The challenge taken up in the present chapter consists in representing some a priori knowledge about unequally spaced tube interfaces in the equivalent LPC lattice filter. This will constitute a particular kind of production constraint, and we will see that it allows to de-couple the dimensionality of the production model from the dimensions of is spectral counterpart.

To begin with, the review of the reasoning linking LPC to lossless tubes will be the occasion for us to extend the classical analogy to the case of tubes with unequal length sections.

2.1 The DSP side of the problem: Linear Prediction Coding of speech

The Linear Prediction Coding method considers that any signal may be modeled as the output of an all-pole filter excited by a white noise or an impulse train source [Makhoul, 1975a] (figure 2.1).

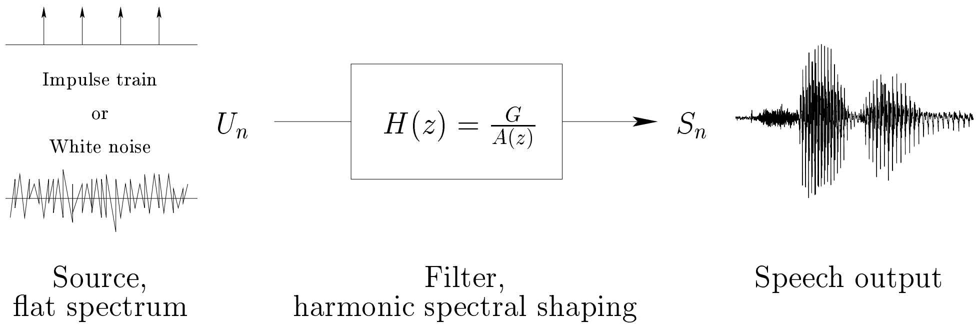


Figure 2.1: The Linear Prediction model of speech production.

The all-pole transfer function has the form:

$$H(z) = \frac{G}{A(z)} \tag{2.1}$$

where G is a gain term and $A(z)$ is a polynomial of the form:

$$A(z) = \sum_{k=0}^{p} a_k z^{-k}; \quad a_0 = 1 \tag{2.2}$$

$A(z)$ is called the *inverse filter*, and p represents the number of poles it contains.

If H(z) is stable [Oppenheim and Schafer, 1989; Kunt, 1996], $A(z)$ can be implemented in a lattice form [Friedlander, 1982] (fig. 2.2). The lattice parameters are the *reflection coefficients* k_i, related to the prediction coefficients a_k by the following recursion:

$$\begin{cases} a_m^{(m)} = k_m \; ; \quad a_0^{(m)} = 1 \\ \\ a_j^{(m)} = a_j^{(m-1)} + k_m a_{m-j}^{(m-1)}; \quad 1 \leq j \leq m-1 \end{cases} \tag{2.3}$$

This recursion allows to grow the degree of $A(z)$ by addition of successive individual lattice cells (growing with $m = 1, 2, \cdots, p$ from $a_0 = k_0 = 1$).

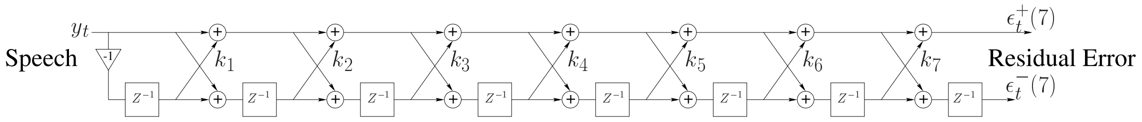


Figure 2.2: The inverse filter in its lattice form. y_t is the speech input, $\epsilon_t^+(i)$ (resp. $\epsilon_t^-(i)$) is the residual forward (resp. backward) residual error after the i^{th} filtering stage.

The lattice form has several advantages over the transverse form :

- it provides a simple test for the stability of the filter : $H(z)$ is stable if $\forall i,\ |k_i| < 1$;
- the transfer function can be grown by a simple serial addition of independent lattice filtering cells. Hence, the values of the reflection coefficients $k_{i,\, i=0,\cdots,m}$, are not modified by the addition of k_{m+1}. Conversely, in the transverse form, all the coefficients $a^{(m)}_{i,\, i=0,\cdots,m}$, are modified if the order of the filter is grown to $(m+1)$;
- the reflection coefficients are known to have a good robustness to quantization errors [Viswanathan and Makhoul, 1975]. Hence, they are used in state of the art coding systems such as the Code Excited Linear Prediction (CELP) [Schroeder and Atal, 1985], used in GSM telephony.

This model represents a somewhat unspecialized production model, in the sense that the only assumption it makes about the represented signal is that it should correspond to the output of an all-pole filtering process. In the DSP terminology, the signal is assimilated to an Auto-Regressive (AR) process.

On the one hand, this assumption is well adapted to the modeling of the voiced parts of speech, which have peaky spectra. With adequate estimators for the coefficients a_i or k_i and a correct guess for the order of the model, LPC also achieves a reasonably good spectral fitting in the non-voiced and nasalized parts of speech.

On the other hand, the underlying model is mainly based on mathematical formulations of spectral or temporal fitting problems [Makhoul, 1975a,b; Markel and Gray, 1976] rather than on a phenomenologic study of speech production. For instance, it describes speech with a number of degrees of freedom corresponding to the LPC order, whereas speech production may result from the action of a more limited number of articulators.

2.1.1 Reducing the number of degrees of freedom of the LPC filter

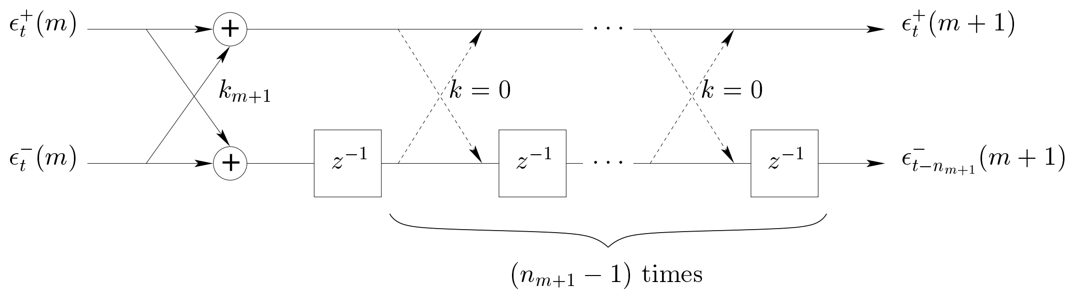
Via the lattice form, the transfer function $H(z) = \frac{1}{A_M(z)} = \frac{1}{\sum_{i=0}^{M} a_i z^{-i}}$ associated with an M^{th} order all-pole model can be built recursively by application of the following matrix recursion [Oppenheim and Schafer, 1989; Haykin, 1996] :

$$\begin{aligned} \begin{bmatrix} A_{m+1}(z) \\ B_{m+1}(z) \end{bmatrix} &= \begin{bmatrix} 1 & k_{m+1} \\ k_{m+1}z^{-1} & z^{-1} \end{bmatrix} \begin{bmatrix} A_m(z) \\ B_m(z) \end{bmatrix} \\ \begin{bmatrix} A_0(z) \\ B_0(z) \end{bmatrix} &= \begin{bmatrix} 1 \\ -z^{-1} \end{bmatrix} \end{aligned} \tag{2.4}$$

where :

- $A_m(z)$ is the transfer function of an m^{th} order forward predictor, modeling the current sample as a linear combination of $(m-1)$ past samples;
- $B_m(z)$ is the transfer function of an m^{th} order backward predictor, modeling the m^{th} past sample as a linear combination of $(m-1)$ future samples;
- k_{m+1} is the reflection coefficient allowing to grow the predictors from order (m) to order $(m+1)$.

Suppose that from the m^{th} step of this recursion, a known number $(n_{m+1}-1)$ of the reflection coefficients following k_{m+1} are fixed to zero. The equivalent lattice flow chart will look like :



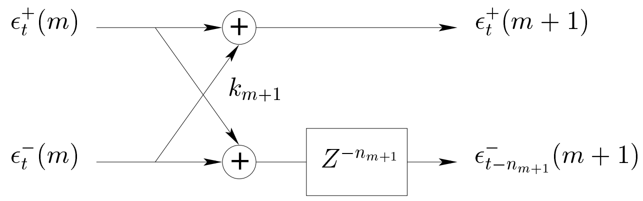
and the corresponding portion of the matrix recursion becomes :

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \cdots \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix}}_{(n_{m+1}-1) \text{ times}} \begin{bmatrix} 1 & k_{m+1} \\ k_{m+1}z^{-1} & z^{-1} \end{bmatrix} \tag{2.5}$$

which can be reduced to a single matrix :

$$\begin{bmatrix} 1 & k_{m+1} \\ k_{m+1}z^{-n_{m+1}} & z^{-n_{m+1}} \end{bmatrix} \tag{2.6}$$

This matrix describes an inverse filtering cell of the form:



Hence, if the recursion steps are re-numbered from 1 to the number of non-zero reflection coefficients (i.e. if the steps with null reflection coefficients are ignored in the indexing), the whole matrix recursion can be rewritten as:

$$\begin{aligned} \begin{bmatrix} A_{m+1}(z) \\ B_{m+1}(z) \end{bmatrix} &= \begin{bmatrix} 1 & k_{m+1} \\ k_{m+1}z^{-n_{m+1}} & z^{-n_{m+1}} \end{bmatrix} \begin{bmatrix} A_m(z) \\ B_m(z) \end{bmatrix} \\ \begin{bmatrix} A_0(z) \\ B_0(z) \end{bmatrix} &= \begin{bmatrix} 1 \\ -z^{-n_0} \end{bmatrix} \end{aligned} \tag{2.7}$$

where the delays $z^{-n_{m+1}}$ can be of any order greater than or equal to 1 for each step of the recursion.

2.1.2 Effects of the constraint in the DSP domain

Generally speaking, all the relations that describe the mathematics of standard lattice filters are still valid in the framework of the constrained lattices described above [Krstulović and Bimbot, 2000]: they will only undergo formal modifications due to the imposition of zero-values at particular places. For instance, the transfer function $A_M(z)$ of the forward-error filter remains a polynom in z^{-1}. Similarly, the backward predictor $B_m(z)$ can be deduced from the forward predictor $A_m(z)$, using the expression:

$$B_m(z) = -z^{-\sum_{i=0}^{m} n_i} A_m(1/z) \tag{2.8}$$

The growth of the forward predictor can thus still be formalized as:

$$A_{m+1}(z) = A_m(z) + k_{m+1} B_m(1/z) \tag{2.9}$$

Nevertheless, the inclusion of the a priori null values introduces interesting structural constraints to the Linear Prediction model.

From equation (2.7), one can remark that the global LPC order of $A_M(z)$ is equal to the sum of the various delays $n_{m,\, m=0,\cdots,M-1}$. In the classical case, where $n_m = 1\ \ \forall m$, the global order is equal to the number of reflection coefficients. Conversely, in the constrained lattice case, *the global order can be greater than the number of unconstrained reflection coefficients.*

As a matter of fact, the reflection coefficients can be viewed as the intrinsic degrees of freedom (DoFs) of the equivalent linear predictor. Constraining some reflection coefficients to stay zero-valued amounts to reducing the intrinsic number of DoFs without changing the global order of the predictor. Hence, the number of DoFs and the global order become independent values. For instance, the transfer function $H(z)$ (and the corresponding spectral model) can have more poles than the number of parameters used to describe the pole locations. Alternately, for a fixed specification of the number of DoFs, a wider portion of the signal's past, corresponding to the greater global order, can be used for the prediction.

Finally, the choice of the location of the zero-valued coefficients allows to represent some structural constraints, pertaining to signal production from unequal-length tubes, in an otherwise unspecialized model.

2.2 The physical side of the problem: acoustic filtering by lossless tubes

This section follows the path that leads from the description of the state of a fluid vibrating in a tube to the computation of the tube's transfer function in the z-transform domain. Unequal length sections are considered instead of the traditional case of a uniform discretization of the tube[1].

2.2.1 Fluid dynamics basis of the problem

Basic system

The vocal tract is considered to be an acoustic tube discretized in M cylindrical sections of *various lengths* (figure 2.3). These sections are numbered in crescent order *from lips to glottis.* Their cross-sectional areas $S_i(t)_{i=1,\cdots,M}$, are allowed to vary across time. Alternately, their lengths l_i, ${}_{i=1,\cdots,M}$, remain fixed.

Simplifying assumptions

- The sound waves are plane fluid waves [Flanagan, 1972, pp.24-25] [Morse and Ingard, 1968, p.467].

- The tube is rigid (no wall impedance is considered).

[1] The same kind of reflection processes are also met in geophysics [Robinson and Durrani, 1986], with layered earth models instead of tubes, and where the earth layers may be unequally spaced.

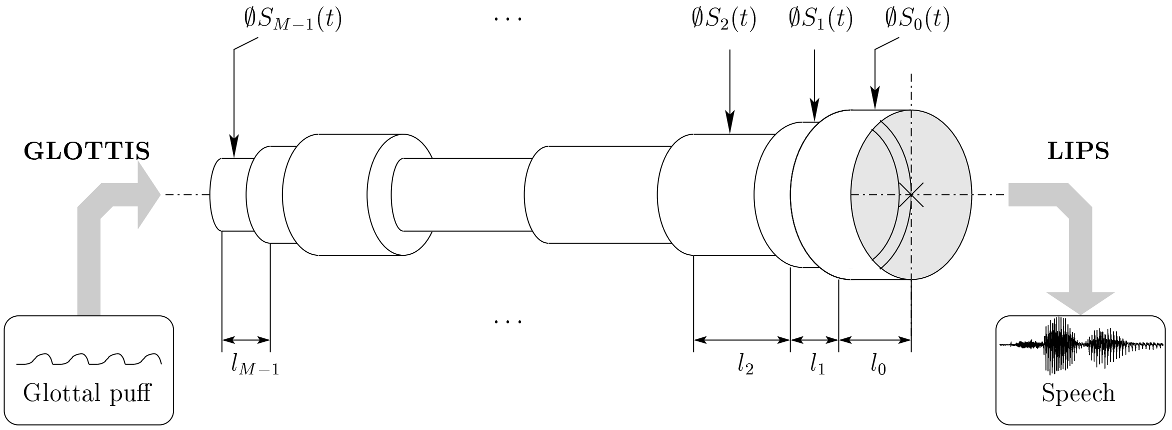


Figure 2.3: An acoustic tube with non-uniform lengths.

- The quantization of the tube introduces an acceptable error if the lengths of the sections are kept short compared to a wavelength at the highest frequency of interest [Flanagan, 1972, p.25] :

$$l_{max} \ll \frac{c}{F_{max}}$$

  where c is the speed of sound, l_{max} is the maximum acceptable length for the longest tube section and F_{max} is an upper bound for the considered acoustic spectrum.

- The energy losses due to the viscosity of the air and to the heat conduction are neglected.

Fluid state equation

Posing the problem in terms of fluid dynamics, we can consider that the volume velocity $u_m(t,d)$ and the pressure $p_m(t,d)$ in the m^{th} section derive from a potential $\Phi_m(t,d)$:

$$\begin{cases} u_m(t,d) = -S_m \frac{\partial \Phi_m(t,d)}{\partial d} \\ \\ p_m(t,d) = \rho \frac{\partial \Phi_m(t,d)}{\partial t} \end{cases} \tag{2.10}$$

where :

- t is the time variable;
- d is the distance variable;
- S_m is the cross-sectional area of the m^{th} section;
- ρ is the density of the air.

The evolution of the fluid state is thereafter described by Webster's equation [Bonder, 1983] :

$$\frac{\partial^2 \Phi_m(t,d)}{\partial d^2} - \frac{1}{c^2} \frac{\partial^2 \Phi_m(t,d)}{\partial t^2} = 0 \tag{2.11}$$

where c denotes the sound velocity.

Equation solving

If we assume that the excitation source (the "glottis" of the tube) delivers a sinusoidal signal, then the solution of the differential equation (2.11) is of the classic form :

$$\Phi_m(t,d) = A\exp^{j\omega(t-d/c)} + B\exp^{j\omega(t+d/c)} \tag{2.12}$$

where A and B are constants. If the excitation signal is made of a linear combination of sine waves, which is the case for signals admitting a Fourier Series development, the corresponding solution is a linear combination of the solutions for any individual sinusoidal component. In the case of speech, the relations developed here do not loose their generality (in the limits of the assumptions made at the beginning) since the non-sinusoidal excitation such as the wave coming out of the vocal cords admits a development in Fourier series.

Injecting the solution (2.12) in (2.10), we have :

$$\left\{ \begin{array}{l} u_m(t,d) = \frac{j\omega S_m A}{c}\exp^{j\omega(t-d/c)} - \frac{j\omega S_m B}{c}\exp^{j\omega(t+d/c)} \\ \\ p_m(t,d) = \rho j\omega A\exp^{j\omega(t-d/c)} + \rho j\omega B\exp^{j\omega(t+d/c)} \end{array} \right. \tag{2.13}$$

In the above equation, we can remark that the volume velocity $u_m(t,d)$ is made of a forward-traveling wave $u_m^+(t,d)$ and a backward-traveling wave $u_m^-(t,d)$:

$$\left\{ \begin{array}{l} u_m^+(t,d) = \frac{j\omega S_m A}{c}\exp^{j\omega(t-d/c)} \\ \\ u_m^-(t,d) = \frac{j\omega S_m B}{c}\exp^{j\omega(t+d/c)} \end{array} \right. \tag{2.14}$$

Hence, the solution (2.13) can be decomposed in the following way :

$$\left\{ \begin{array}{l} u_m(t,d) = u_m^+(t,d) - u_m^-(t,d) \\ \\ p_m(t,d) = \frac{\rho c}{S_m}\left\{u_m^+(t,d) + u_m^-(t,d)\right\} \end{array} \right. \tag{2.15}$$

Additional continuity relations

At the connection between the section m and the section $m-1$, the volume velocity and the pressure must be continuous. We therefore have the additional relations :

$$\left\{ \begin{array}{l} u_m(t,d_m) = u_{m-1}(t,d_m) \\ \\ p_m(t,d_m) = p_{m-1}(t,d_m) \end{array} \right. \tag{2.16}$$

d_m being the distance between the glottis and the connection of the sections m and $m-1$ (see figure 2.4).

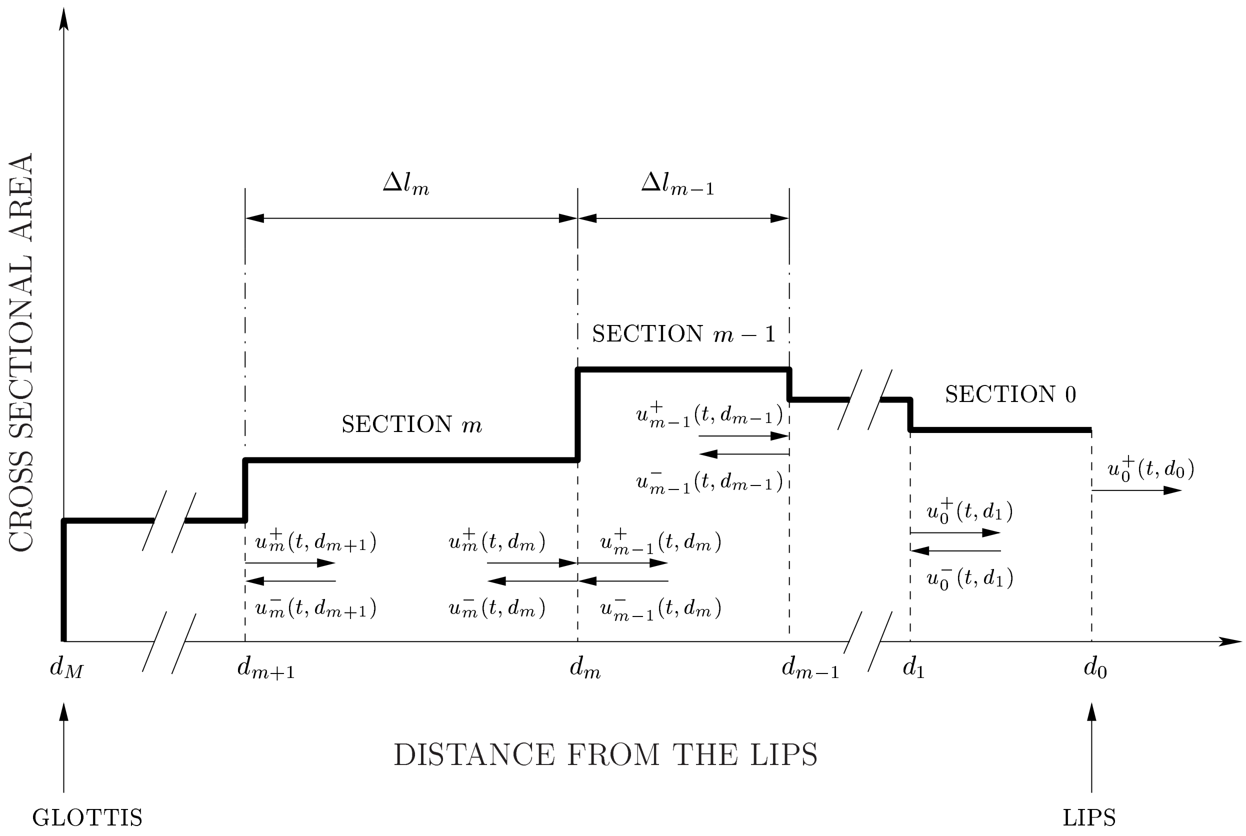


Figure 2.4: Volume velocities in the vocal tract modeled as a non-uniform lossless acoustic tube.

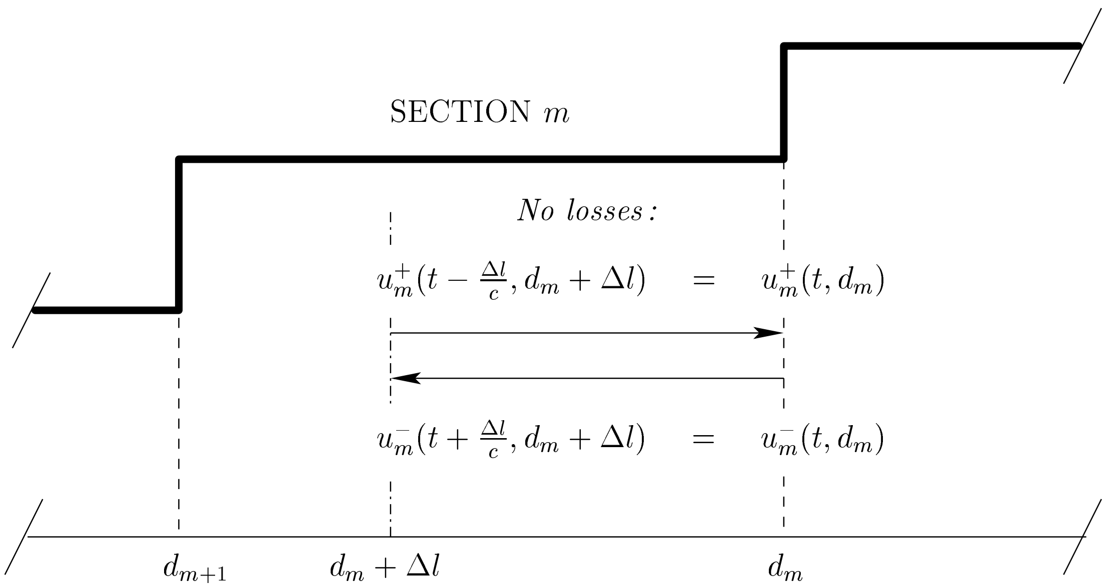


Figure 2.5: Conservation of the volume velocities in a given tube section.

Alternately, since there is no loss in a particular section, we also have *inside the limits of a section*:

$$\left\{ \begin{array}{l} u_m^+(t, d_m) = u_m^+(t - \frac{\Delta l}{c}, d_m + \Delta l) \\ \\ u_m^-(t, d_m) = u_m^-(t + \frac{\Delta l}{c}, d_m + \Delta l) \end{array} \right. \tag{2.17}$$

Δl being a time-independent arbitrary length taken inside the considered piece of tube and c being the speed of sound (figure 2.5). This equation system tells us that the forward traveling wavefront standing at the abscissa d_m at time t was standing at the abscissa $d_m + \Delta l$ at an earlier time $t - \frac{\Delta l}{c}$. Similarly, the backward traveling wavefront which stands at the abscissa d_m at time t will reach the abscissa $d_m + \Delta l$ at a later time $t + \frac{\Delta l}{c}$. In particular, when starting from the abscissa d_m to reach the abscissa d_{m-1}, we have :

$$\left\{ \begin{array}{lcl} u_m^+(t, d_m) & = & u_m^+(t - \frac{\Delta l_m}{c}, d_m + \Delta l_m) \\ & = & u_m^+(t - \frac{\Delta l_m}{c}, d_{m+1}) \\ \\ u_m^-(t, d_m) & = & u_m^-(t + \frac{\Delta l_m}{c}, d_m + \Delta l_m) \\ & = & u_m^-(t + \frac{\Delta l_m}{c}, d_{m+1}) \end{array} \right. \tag{2.18}$$

From (2.15) and (2.16), it results that :

$$\left\{ \begin{array}{l} u_m^+(t - \frac{\Delta l_m}{c}, d_{m+1}) - u_m^-(t + \frac{\Delta l_m}{c}, d_{m+1}) = u_{m-1}^+(t, d_m) - u_{m-1}^-(t, d_m) \\ \\ \frac{\rho c}{S_m} \left\{ u_m^+(t - \frac{\Delta l_m}{c}, d_{m+1}) + u_m^-(t + \frac{\Delta l_m}{c}, d_{m+1}) \right\} = \frac{\rho c}{S_{m-1}} \left\{ u_{m-1}^+(t, d_m) + u_{m-1}^-(t, d_m) \right\} \end{array} \right. \tag{2.19}$$

In the above system, the spatial dimension d_m is completely specified by the interface index m. Furthermore, from (2.17), it is clear that the fluid state in a whole section can be deduced from a value measured at the preceding junction. Hence, the relations (2.19) can be restricted to the values at the junctions without a loss of generality, and the distance variables can be dropped :

$$\left\{ \begin{array}{l} u_m^+(t - \frac{\Delta l_m}{c}) - u_m^-(t + \frac{\Delta l_m}{c}) = u_{m-1}^+(t) - u_{m-1}^-(t) \\ \\ \frac{\rho c}{S_m} \left\{ u_m^+(t - \frac{\Delta l_m}{c}) + u_m^-(t + \frac{\Delta l_m}{c}) \right\} = \frac{\rho c}{S_{m-1}} \left\{ u_{m-1}^+(t) + u_{m-1}^-(t) \right\} \end{array} \right. \tag{2.20}$$

Since the speed of sound is constant, we can note $\Delta t_m = \frac{\Delta l_m}{c}$ the time necessary for a wave to travel through a tube of length Δl_m, and we finally get :

$$\left\{ \begin{array}{l} u_m^+(t - \Delta_m t) - u_m^-(t + \Delta_m t) = u_{m-1}^+(t) - u_{m-1}^-(t) \\ \\ u_m^+(t - \Delta_m t) + u_m^-(t + \Delta_m t) = \frac{S_m}{S_{m-1}} \left\{ u_{m-1}^+(t) + u_{m-1}^-(t) \right\} \end{array} \right. \tag{2.21}$$

The above steps are very important, because they allow to express the spatio-temporal description of the whole fluid state in terms of time series measured at the tube junctions. Furthermore, the junctions can be located by discrete time positions rather than discrete distances. These considerations open the access to a discrete signal processing analysis framework.

2.2.2 From fluids to discrete signals

"Folding" of the filtering process and appearance of the reflection coefficients

Defining some reflection coefficients as:

$$\mu_m = \frac{S_m - S_{m-1}}{S_m + S_{m-1}} \tag{2.22}$$

and applying to (2.21) when solving for u_m^+ and u_m^-, we obtain:

$$\begin{cases} u_m^+(t - \Delta_m t) = \frac{1}{1-\mu_m} \left\{ u_{m-1}^+(t) + \mu_m u_{m-1}^-(t) \right\} \\ \\ u_m^-(t + \Delta_m t) = \frac{1}{1-\mu_m} \left\{ \mu_m u_{m-1}^+(t) + u_{m-1}^-(t) \right\} \end{cases} \tag{2.23}$$

Application of the z-transform

To be able to apply the z-transform to (2.23), and thus definitely join the DSP world, we need to define a unit length Δl_{unit} which is the greatest common divisor of the set of lengths $\{\Delta l_m, m = 1, \cdots, M\}$. This unit length allows to express the length of each individual section as a non-dimensional integer number n_m, via the relation:

$$n_m = \frac{\Delta l_m}{\Delta l_{unit}} \tag{2.24}$$

Since the speed of sound is constant, the unit length relates to a time constant $\Delta_{unit} t$ such that:

$$n_m = \frac{\Delta t_m}{\Delta t_{unit}} \tag{2.25}$$

If we sample the temporal signals with a frequency of $F_s = \frac{c}{2\Delta l_{unit}}$, we can apply the z-transform to (2.23), with z defined as $z = e^{j\omega 2\Delta l_{unit}/c} = e^{j\omega 2\Delta t_{unit}}$, and we obtain:

$$\begin{cases} z^{\frac{-n_m}{2}} U_m^+(z) = \frac{1}{1-\mu_m} \left[U_{m-1}^+(z) + \mu_m U_{m-1}^-(z) \right] \\ \\ z^{\frac{n_m}{2}} U_m^-(z) = \frac{1}{1-\mu_m} \left[\mu_m U_{m-1}^+(z) + U_{m-1}^-(z) \right] \end{cases} \tag{2.26}$$

i.e.:

$$\begin{cases} U_m^+(z) = \frac{z^{\frac{n_m}{2}}}{1-\mu_m} \left[U_{m-1}^+(z) + \mu_m \, U_{m-1}^-(z) \right] \\ \\ U_m^-(z) = \frac{z^{\frac{n_m}{2}}}{1-\mu_m} \left[\mu_m \, z^{-n_m} \, U_{m-1}^+(z) + z^{-n_m} \, U_{m-1}^-(z) \right] \end{cases} \tag{2.27}$$

and, in matrix notation:

$$\begin{bmatrix} U_m^+(z) \\ U_m^-(z) \end{bmatrix} = \frac{z^{\frac{n_m}{2}}}{1-\mu_m} \begin{bmatrix} 1 & \mu_m \\ \mu_m z^{-n_m} & z^{-n_m} \end{bmatrix} \begin{bmatrix} U_{m-1}^+(z) \\ U_{m-1}^-(z) \end{bmatrix} \tag{2.28}$$

Development of the transfer function

If we assume that the section corresponding to the lips extremity is connected to a tube of infinite section, the following boundary condition at the front end (or lips end) of our model is verified:

$$S_{-1} = \infty \;\Rightarrow\; \mu_0 = -1$$

Applying this condition, we can write:

$$\begin{bmatrix} U_m^+(z) \\ U_m^-(z) \end{bmatrix} = z^{\left(\frac{1}{2}\sum_{k=0}^{m} n_k\right)}\, K_m \begin{bmatrix} D_m^+(z) \\ D_m^-(z) \end{bmatrix} \left\{U_0^+(z) - U_0^-(z)\right\} \tag{2.29}$$

with

$$\begin{bmatrix} D_m^+(z) \\ D_m^-(z) \end{bmatrix} = \begin{bmatrix} 1 & \mu_m \\ \mu_m z^{-n_m} & z^{-n_m} \end{bmatrix} \begin{bmatrix} 1 & \mu_{m-1} \\ \mu_{m-1} z^{-n_{m-1}} & z^{-n_{m-1}} \end{bmatrix} \cdots \begin{bmatrix} 1 & \mu_1 \\ \mu_1 z^{-n_1} & z^{-n_1} \end{bmatrix} \begin{bmatrix} 1 \\ -z^{-n_0} \end{bmatrix} \tag{2.30}$$

and

$$K_m = \prod_{i=0}^{m} \frac{1}{1-\mu_i} \tag{2.31}$$

Neglecting the overall delay $z^{\left(\frac{1}{2}\sum_{k=0}^{m} n_k\right)}$ and the gain K_m, *the true transfer function for the volume velocity when traveling from lips to glottis corresponds to* $D_m^+(z)$, *and can be built recursively by applying:*

$$\begin{aligned} \begin{bmatrix} D_{m+1}^+(z) \\ D_{m+1}^-(z) \end{bmatrix} &= \begin{bmatrix} 1 & \mu_{m+1} \\ \mu_{m+1} z^{-n_{m+1}} & z^{-n_{m+1}} \end{bmatrix} \begin{bmatrix} D_m^+(z) \\ D_m^-(z) \end{bmatrix} \\ \begin{bmatrix} D_0^+(z) \\ D_0^-(z) \end{bmatrix} &= \begin{bmatrix} 1 \\ -z^{-n_0} \end{bmatrix} \end{aligned} \tag{2.32}$$

As a matter of fact, the recursion (2.32) is exactly the same as the recursion (2.7) found in the section 2.1.1 to grow the transfer function of a constrained lattice filter.

After the development of this recursion up to the order M (i.e., for a tube with M sections), $D_M^+(z)$ will represent the transfer function describing the waveform $U_M^+(z)$, entering the glottal end, as a function of the waveform $\left\{U_0^+(z) - U_0^-(z)\right\}$ output at the lips end. The "synthesis oriented" transfer function, describing the forward filtering action of the vocal tract, corresponds simply to the inverse of $D_M^+(z)$:

$$H_M^+(z) = \frac{1}{D_M^+(z)} \tag{2.33}$$

Alternately, $D_M^-(z)$ describes the waveform $U_M^-(z)$, reflected towards the glottal end, as a function of the waveform $\left\{U_0^+(z) - U_0^-(z)\right\}$ output at the lips end. It is not of direct interest in a speech synthesis framework, since the backward wave is usually considered to be absorbed by the lungs cavity. Nevertheless, this "backward" transfer function is useful from an analysis point of view. We can show by mathematical induction from equation (2.30) that we have, for any step m:

$$D_m^-(z) = -z^{-\sum_{k=0}^{m} n_k} D_m^+(1/z) \tag{2.34}$$

This means that the forward inverse filtering function $D_m^+(z)$ is always the time shifted reciprocal of the backward inverse filtering function $D_m^-(z)$. Developing (2.32) for the order $m+1$, and applying (2.34), we obtain:

$$\begin{cases} D_{m+1}^+(z) & = & D_m^+(z) \ - \ \mu_{m+1}\, z^{-\sum_{k=0}^{m+1} n_k}\, D_m^+(1/z) \\ & & \\ D_{m+1}^+(1/z) & = & -\mu_{m+1}\, z^{\sum_{k=0}^{m+1} n_k}\, D_m^+(z) \ + \ D_m^+(1/z) \end{cases} \tag{2.35}$$

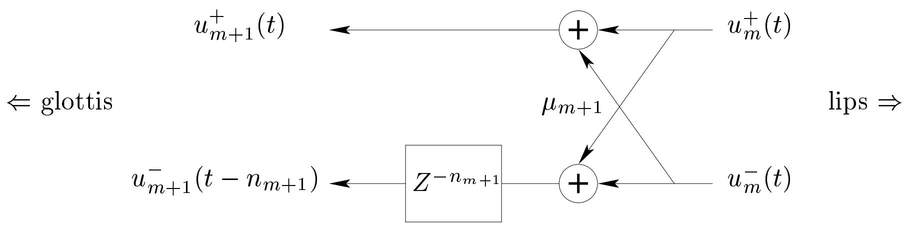
If we change the variable z to $1/z$ in the first of the above formulae, we obtain the formula in the second line. Both formulae describe equivalently the relationship between $D_{m+1}^+(z)$ and $D_m^+(z)$. Starting from $D_0^+(z) = 1$, the recursive application of this relationship describes the growth of the transfer function when one adds a section to the acoustic tube. This equation represents an explicit development of the matricial recursion (2.32).

2.2.3 Lattice form of the acoustic filtering process in the general case

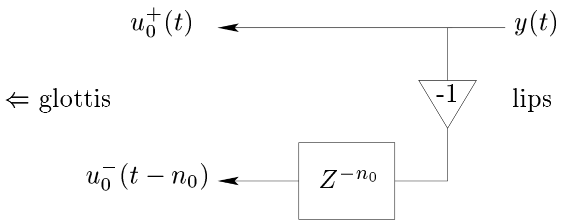
In the equation (2.32), each of the elementary matrix blocks of the form

$$\begin{bmatrix} 1 & \mu_{m+1} \\ \mu_{m+1} z^{-n_{m+1}} & z^{-n_{m+1}} \end{bmatrix} \tag{2.36}$$

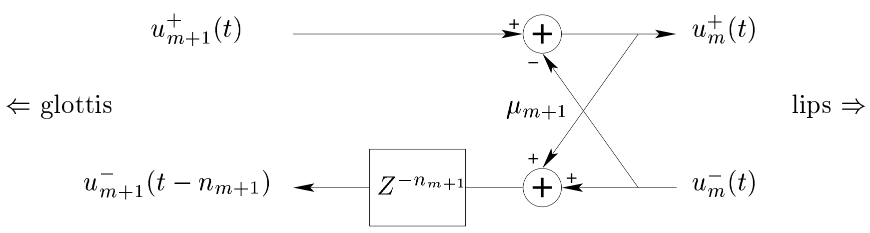
corresponds to an inverse filtering cell of the form:



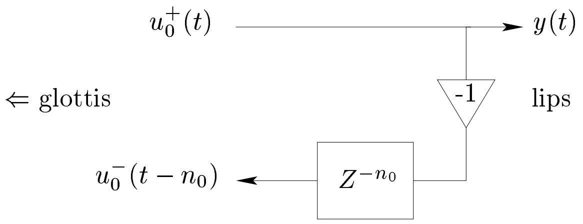
The inverse transfer function $D_M^+(z)$ can thus be built by connecting M of these cells serially, starting with a first cell of the form:



Alternately, the corresponding synthesis filter with a transfer function $H(z) = 1/D_M^+(z)$ can be represented by a series of cells of the form:



with a first cell of the form:



2.2.4 Nature of the acoustic filtering process in the general case

We will now develop the relations (2.35) in order to study more precisely the nature and the growth of the transverse form of the transfer function $H(z) = \frac{1}{D_M^+(z)}$. This development will be made in the traditional case of tubes with equal length sections, corresponding to unit delays, and then in the more general case of tubes with unequal-length sections, corresponding to delays of any order.

Transverse form in the classical case: the length of the sections is uniform

In this case, the transmission delay induced in every piece of tube is the same. It corresponds to $n_k = 1 \quad \forall k$, i.e. $z^{-n_k} = z^{-1} \quad \forall k$ in all the above equations. z is then defined as $z = e^{j\omega 2\Delta l/c}$, Δl being the length of every cylindrical piece of tube.

From equation (2.30), it can be seen that $D_m^+(z)$ is of the form:

$$D_m^+(z) = \sum_{i=0}^{m} a_i^{(m)} z^{-i} \tag{2.37}$$

which corresponds to a Finite Impulse Response (FIR) inverse filter.

When used for acoustic filtering in the "synthesis direction" (from glottis to lips), it therefore acts as an all-pole filter:

$$H(z) = \frac{1}{D_m^+(z)} = \frac{1}{\sum_{i=0}^{m} a_i^{(m)} z^{-i}} \tag{2.38}$$

Hence, the lossless tube model represents speech production as an all-pole filtering process applied to the glottal waveform, or, equivalently, as a Linear Prediction process.

When adding a new cylindrical section to our tube, we have:

$$D_{m+1}^+(z) = \sum_{i=0}^{m+1} a_i^{(m+1)} z^{-i} \tag{2.39}$$

which means that the degree of the polynomial inverse transfer function increases by one at each tube growing step (see for instance the development operated in figure 2.6).

In this case, the relation between the transfer function coefficients a_k and the reflection coefficients μ_i can be formalized explicitly. The relation (2.35) gives:

$$\sum_{i=0}^{m+1} a_i^{(m+1)} z^{-i} = \sum_{i=0}^{m} a_i^{(m)} z^{-i} + \mu_{m+1}\, z^{-(m+1)} \sum_{i=0}^{m} a_i^{(m)} z^{i} \tag{2.40}$$

i.e.

$$\sum_{i=0}^{m+1} a_i^{(m+1)} z^{-i} = \sum_{i=0}^{m} a_i^{(m)} z^{-i} + \mu_{m+1} \sum_{i=0}^{m} a_i^{(m)} z^{i-(m+1)} \tag{2.41}$$

or, changing the mute index i to $(m+1-i)$ in the second sum of the right term:

$$\sum_{i=0}^{m+1} a_i^{(m+1)} z^{-i} = \sum_{i=0}^{m} a_i^{(m)} z^{-i} + \mu_{m+1} \sum_{i=1}^{m+1} a_{m+1-i}^{(m)} z^{-i} \tag{2.42}$$

Identifying the coefficients of the polynoms in z^{-i} on each side of the equal sign, we obtain:

$$\left\{ \begin{array}{rcl} a_0^{(m+1)} & = & a_0^{(m)} = 1 \\ a_1^{(m+1)} & = & a_1^{(m)} + \mu_{m+1} a_m^{(m)} \\ & \vdots & \\ a_m^{(m+1)} & = & a_m^{(m)} + \mu_{m+1} a_1^{(m)} \\ a_{m+1}^{(m+1)} & = & \mu_{m+1} a_0^{(m)} \end{array} \right. \tag{2.43}$$

$$\begin{bmatrix} 1 \\ -z^{-1} \end{bmatrix}$$

$$\begin{bmatrix} 1 & - & \mu_1 z^{-1} \\ \mu_1 z^{-1} & - & z^{-2} \end{bmatrix}$$

$$\begin{bmatrix} 1 & + & (\mu_2\mu_1 - \mu_1) z^{-1} & - & \mu_2 z^{-2} \\ \mu_2 z^{-1} & + & (\mu_1 - \mu_2\mu_1) z^{-2} & - & z^{-3} \end{bmatrix}$$

$$\begin{bmatrix} 1 & + & (\mu_2\mu_1 - \mu_1 + \mu_3\mu_2) z^{-1} & + & (-\mu_2 + \mu_3\mu_1 - \mu_3\mu_2\mu_1) z^{-2} & - & \mu_3 z^{-3} \\ \mu_3 z^{-1} & + & (\mu_2 - \mu_3\mu_1 + \mu_3\mu_2\mu_1) z^{-2} & + & (\mu_1 - \mu_2\mu_1 - \mu_3\mu_2) z^{-3} & - & z^{-4} \end{bmatrix}$$

$$\begin{bmatrix} 1 & + & \frac{\mu_2\mu_1 - \mu_1 + \mu_4\mu_3 + \mu_3\mu_2}{z} & + & \frac{-\mu_2 + \mu_4\mu_3\mu_2\mu_1 - \mu_4\mu_3\mu_1 + \mu_3\mu_1 - \mu_3\mu_2\mu_1 + \mu_4\mu_2}{z^2} & + & \frac{-\mu_4\mu_3\mu_2 - \mu_3 + \mu_4\mu_1 - \mu_4\mu_1\mu_2}{z^3} & - & \mu_4 z^{-4} \\ \mu_4 z^{-1} & + & \frac{\mu_4\mu_3\mu_2 + \mu_3 - \mu_4\mu_1 + \mu_4\mu_1\mu_2}{z^2} & + & \frac{\mu_2 - \mu_4\mu_3\mu_2\mu_1 + \mu_4\mu_3\mu_1 - \mu_3\mu_1 + \mu_3\mu_2\mu_1 - \mu_4\mu_2}{z^3} & + & \frac{\mu_1 - \mu_2\mu_1 - \mu_4\mu_3 - \mu_3\mu_2}{z^4} & - & z^{-5} \end{bmatrix}$$

$$\vdots$$

Figure 2.6: **Growth of the transfer function of a tube with equal length sections.** Note the regular increase in the polynomial degrees.

which can be formalized in a simple form as:

$$\left|\begin{array}{ll} \text{Initialization:} & a_0^{(0)} = 1 \quad \text{and} \quad a_k^{(0)} = 0, \ \forall k \neq 0 \\ \text{Recursion:} & \begin{cases} a_0^{(m+1)} = 1 \\ a_i^{(m+1)} = a_i^{(m)} - \mu_{m+1} a_{m+1-i}^{(m)} \ ; \ 1 \leq i \leq m \\ a_{m+1}^{(m+1)} = \mu_{m+1} \end{cases} \end{array}\right. \tag{2.44}$$

These equations are identical to those linking the reflection coefficients k_i to the prediction coefficients a_i in the classical definition of the lattice filter (see equation (2.3), page 12).

Transverse form in the more general case: the length of the sections is not uniform

In the more general case where the length of the sections is not uniform, we must deal with the irregular delays and the z^{-n_k} not being equal to z^{-1}. To formalize the growth of the transfer function in a readable way, we will borrow the notation of the summation indexes to the set theory.

Let Ω_m be the set of all possible indexes k for the discrete delays n_k met in a particular tube:

$$\Omega_m = \{0, 1, \cdots, m\} \tag{2.45}$$

Let Γ be a set containing one of the possible index combinations[2]:

$$\Gamma \subset \Omega_m \tag{2.46}$$

We know from equation (2.30) that $D_m^+(z)$ is a polynomial in z and we can now express its form as:

$$D_m^+(z) = \sum_{\Gamma \subset \{0,1,\cdots m\}} a_\Gamma^{(m)} z^{-\sum_{k \in \Gamma} n_k} \tag{2.47}$$

This transfer function has a special (constrained) form: it still corresponds to a FIR filter (still all-pole when reverted for synthesis), *but not all the polynomial degrees are represented.*

[2] Γ belongs to the set of all subsets of Ω_m. We can remark that this later set defines a σ-algebra on the set of delays n_k. A measure on this set could be defined as $\sum_{k \in \Gamma} n_k$. We don't know if such measure theory notions have already been used in the framework of polynomial transfer functions analysis, but researchers interested in measure theory might find here a lead to an alternate way of formalizing the problem.

As before, we can observe the growth of the inverse transfer function. When adding a new section, we have :

$$D^{+}_{m+1}(z) = \sum_{\Gamma \subset \{0,1,\cdots m+1\}} a_{\Gamma}^{(m+1)} z^{-\sum_{k\in\Gamma} n_k} \tag{2.48}$$

We see here that when going from step (m) to step $(m+1)$, the polynomial degree increases by $-n_{m+1}$, and that not all the degrees between 0 and $-\sum_{k\in\Gamma} n_k$ are guaranteed to be represented.

Coming back to the development of the $a_k \Leftrightarrow \mu_i$ relation, the equation (2.35) now gives :

$$\sum_{\Gamma\subset\Omega_{m+1}} a_{\Gamma}^{(m+1)} z^{-\sum_{k\in\Gamma} n_k} = \sum_{\Gamma\subset\Omega_m} a_{\Gamma}^{(m)} z^{-\sum_{k\in\Gamma} n_k} + \mu_{m+1} z^{-\sum_{k\in\Omega_{m+1}} n_k} \sum_{\Gamma\subset\Omega_m} a_{\Gamma}^{(m)} z^{\sum_{k\in\Gamma} n_k} \tag{2.49}$$

i.e.

$$\sum_{\Gamma\subset\Omega_{m+1}} a_{\Gamma}^{(m+1)} z^{-\sum_{k\in\Gamma} n_k} = \sum_{\Gamma\subset\Omega_m} a_{\Gamma}^{(m)} z^{-\sum_{k\in\Gamma} n_k} + \mu_{m+1} \sum_{\Gamma\subset\Omega_m} a_{\Gamma}^{(m)} z^{\sum_{k\in\Gamma} n_k - \sum_{k\in\Omega_{m+1}} n_k} \tag{2.50}$$

For a particular subset Γ of our index set Ω_m, we can show the following :

$$\begin{aligned} \textstyle\sum_{k\in\Gamma} n_k - \sum_{k\in\Omega_{m+1}} n_k &= \underbrace{\sum_{k\in\Gamma} n_k - \sum_{k\in\Omega_m} n_k} \quad -n_{m+1} \\ &= \quad -\textstyle\sum_{k\in\overline{\Gamma}} n_k \quad\quad -n_{m+1} \end{aligned} \tag{2.51}$$

$\overline{\Gamma}$ being the complementary set of Γ so that $\Gamma \cup \overline{\Gamma} = \Omega_m$. Equation (2.50) then becomes :

$$\sum_{\Gamma\subset\Omega_{m+1}} a_{\Gamma}^{(m+1)} z^{-\sum_{k\in\Gamma} n_k} = \sum_{\Gamma\subset\Omega_m} a_{\Gamma}^{(m)} z^{-\sum_{k\in\Gamma} n_k} + \mu_{m+1} \sum_{\Gamma\subset\Omega_m} a_{\Gamma}^{(m)} z^{-\sum_{k\in\overline{\Gamma}} n_k + n_{m+1}} \tag{2.52}$$

In this case, the analytical identification of the polynomial coefficients has to be performed on a case-by-case basis.

This can be illustrated by an example. Let us consider a tube that has 8 sections of unequal lengths with $\Delta l_{unit} = L/30$ (L being the total length of the full tube). The lengths of the sections are distributed as follows from lips to glottis : $\Delta l_0 = 3\Delta l_{unit}$, $\Delta l_1 = 2\Delta l_{unit}$, $\Delta l_2 = 4\Delta l_{unit}$, $\Delta l_3 = 6\Delta l_{unit}$, $\Delta l_4 = 6\Delta l_{unit}$, $\Delta l_5 = 4\Delta l_{unit}$, $\Delta l_6 = 2\Delta l_{unit}$, $\Delta l_7 = 3\Delta l_{unit}$[3]. In this case, the recursion (2.32) expands as follows :

$$\begin{bmatrix} D^{+}(z) \\ D^{-}(z) \end{bmatrix} = \begin{bmatrix} 1 & \mu_7 \\ \mu_7 z^{-3} & z^{-3} \end{bmatrix} \begin{bmatrix} 1 & \mu_6 \\ \mu_6 z^{-2} & z^{-2} \end{bmatrix} \cdots \begin{bmatrix} 1 & \mu_2 \\ \mu_2 z^{-4} & z^{-4} \end{bmatrix} \begin{bmatrix} 1 & \mu_1 \\ \mu_1 z^{-2} & z^{-2} \end{bmatrix} \begin{bmatrix} 1 \\ -z^{-3} \end{bmatrix} \tag{2.53}$$

[3]This model corresponds to the configuration of the Distinctive Regions and Modes phonetic model [Mrayati et al., 1988; Krstulović, 1996].

When observing the growth of the inverse transfer function between, for instance, step 3 and step 4 (see the equations developed in figure 2.7), and replacing the Γ indexes by some integer indexes corresponding to the place of the increasing negative powers of z, we obtain the following set of equations :

$$\left\{\begin{array}{lclcl}
a_0^{(4)} & = & a_0^{(3)} & & = 1 \\
a_1^{(4)} & = & a_1^{(3)} & & \\
a_2^{(4)} & = & a_2^{(3)} & & \\
a_3^{(4)} & = & a_3^{(3)} & & \\
a_4^{(4)} & = & a_4^{(3)} & & \\
a_5^{(4)} & = & a_5^{(3)} & + & \mu_4 a_7^{(3)} \\
a_6^{(4)} & = & a_6^{(3)} & & \\
a_7^{(4)} & = & & & \mu_4 a_6^{(3)} \\
a_8^{(4)} & = & a_7^{(3)} & + & \mu_4 a_5^{(3)} \\
a_9^{(4)} & = & & & \mu_4 a_4^{(3)} \\
a_{10}^{(4)} & = & & & \mu_4 a_3^{(3)} \\
a_{11}^{(4)} & = & & & \mu_4 a_2^{(3)} \\
a_{12}^{(4)} & = & & & \mu_4 a_1^{(3)} \\
a_{13}^{(4)} & = & & & \mu_4
\end{array}\right. \tag{2.54}$$

Therefore, in the general case, it is clear that if we try to operate a polynomial coefficients identity starting from equation (2.35), the relation tying the prediction coefficients a_i and the reflection coefficients μ_i appears as a shifted version of the relation expressed in the equal-length case.

2.3 Summary and key relations

Given some simplifying assumptions in the physical domain (section 2.2.1), the filtering process operated by a lossless acoustic tube can be related to the LPC formalism. This relation is based on the core idea that since the tube portions are lossless, the only information changes that are relevant to the tube filtering action occur at the interfaces of the independent sections. Hence, when using an adequate spatial discretization of the tube and a corresponding adequate time sampling to measure the wave propagation, the spatio-temporal model becomes analogous to a non-dimensional lattice model.

$$\begin{bmatrix} 1 \\ -z^{-3} \end{bmatrix}$$

$$\begin{bmatrix} 1 & - & \mu_1 z^{-3} \\ \mu_1 z^{-2} & - & z^{-5} \end{bmatrix}$$

$$\begin{bmatrix} 1 & + & (\mu_2\mu_1)\, z^{-2} & - & \mu_1 z^{-3} & - & \mu_2 z^{-5} \\ \mu_2 z^{-4} & + & \mu_1 z^{-6} & - & (\mu_2\mu_1)\, z^{-7} & - & z^{-9} \end{bmatrix}$$

$$\begin{bmatrix} 1 & + & \frac{\mu_2\mu_1}{z^2} & - & \frac{\mu_1}{z^3} & + & \frac{\mu_3\mu_2}{z^4} & - & \frac{\mu_2}{z^5} & + & \frac{\mu_3\mu_1}{z^6} & - & \frac{\mu_3\mu_2\mu_1}{z^7} & - & \mu_3 z^{-9} \\ \mu_3 z^{-6} & + & \frac{\mu_3\mu_2\mu_1}{z^8} & - & \frac{\mu_3\mu_1}{z^9} & + & \frac{\mu_2}{z^{10}} & - & \frac{\mu_3\mu_2}{z^{11}} & + & \frac{\mu_1}{z^{12}} & - & \frac{\mu_2\mu_1}{z^{13}} & - & z^{-15} \end{bmatrix}$$

$$\begin{bmatrix} 1 & + & \frac{\mu_2\mu_1}{z^2} - \frac{\mu_1}{z^3} + \frac{\mu_3\mu_2}{z^4} - \frac{\mu_2}{z^5} + \frac{\mu_4\mu_3+\mu_3\mu_1}{z^6} - \frac{\mu_3\mu_2\mu_1}{z^7} + \frac{\mu_4\mu_3\mu_2\mu_1}{z^8} - \frac{\mu_4\mu_3\mu_1+\mu_3}{z^9} + \frac{\mu_4\mu_2}{z^{10}} - \frac{\mu_4\mu_3\mu_2}{z^{11}} + \frac{\mu_4\mu_1}{z^{12}} - \frac{\mu_4\mu_2\mu_1}{z^{13}} & - & \mu_4 z^{-15} \\ \mu_4 z^{-6} & + & \frac{\mu_4\mu_2\mu_1}{z^8} - \frac{\mu_4\mu_1}{z^9} + \frac{\mu_4\mu_3\mu_2}{z^{10}} - \frac{\mu_4\mu_2}{z^{11}} + \frac{\mu_4\mu_3\mu_1+\mu_3}{z^{12}} - \frac{\mu_4\mu_3\mu_2\mu_1}{z^{13}} + \frac{\mu_3\mu_2\mu_1}{z^{14}} - \frac{\mu_4\mu_3+\mu_3\mu_1}{z^{15}} + \frac{\mu_2}{z^{16}} - \frac{\mu_3\mu_2}{z^{17}} + \frac{\mu_1}{z^{18}} - \frac{\mu_2\mu_1}{z^{19}} & - & z^{-21} \end{bmatrix}$$

$$\vdots$$

Figure 2.7: **Growth of the transfer function of a tube with non-uniform length sections.** Note the disturbance in the growth of the polynomial degrees, as compared to figure 2.6.

In particular, the correspondence appears in the matricial recursion used to compute the transfer function both in the physical domain and in the DSP domain, and which has the form:

$$\begin{bmatrix} A_{m+1}(z) \\ B_{m+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & k_{m+1} \\ k_{m+1}z^{-n_{m+1}} & z^{-m+1} \end{bmatrix} \begin{bmatrix} A_m(z) \\ B_m(z) \end{bmatrix}$$

$$\begin{bmatrix} A_0(z) \\ B_0(z) \end{bmatrix} = \begin{bmatrix} 1 \\ -z^{-1} \end{bmatrix} \tag{2.55}$$

The reflection coefficients k_i can be related to the prediction coefficients a_i (and thus to a spectral model) by accounting for some forced zero values and reorganizing the indexing in the classical $k_i \Leftrightarrow a_i$ relation given by:

$$\begin{cases} a_0^{(m)} = 1 \\ a_j^{(m)} = a_j^{(m-1)} + k_m a_{m-j}^{(m-1)}; \quad 1 \leq j \leq m-1 \\ a_m^{(m)} = k_m \end{cases} \tag{2.56}$$

Another key relation is the one linking the parameters of the lattice representation to the physical dimensions of the tube:

$$k_{i+1} = \frac{S_{i+1} - S_i}{S_{i+1} + S_i} \tag{2.57}$$

This relation constitutes a bridge between the physical tube model and the LPC acoustic model, via the parameters of the area function.

In this chapter, we have reviewed the LPC/non-uniform lossless tubes analogy from a generative point of view. Starting from some physical constraints imposed to a lossless tube, namely the unequal length sections, we have studied a method to compute the tube's transfer function, which accounts for its acoustic filtering effects. The development of this method has used the formalism of lattice filters, related to the general LPC framework.

Now, the question arises as how to use this parametric model as an analysis tool, i.e. how to estimate the parameters of the constrained model from the sound of speech.

Chapter 3

Speech analysis with Non-Uniform Topology predictors

In the previous chapter, we have derived a class of parametric production models from the introduction of unequal lengths in the lossless tube models. We have shown that this is equivalent to using non-unit delays at a-priori locations in the lattice all-pole filtering framework. This idea stemmed from the consideration that some well-known speech production models, such as the DRM [Mrayati et al., 1988] model, use unequal lengths in the area functions that they represent. In the present chapter, we will explore the use of the unequal-length models as an analysis tool incorporating specific production constraints.

The idea of using unequal-length models for synthesis or analysis may not be new :

- Frank and Lacroix [1986], following Lacroix and Makai [1979], have proposed to use unequal-length models for the copy-synthesis of speech signals, with the ultimate goal of finding articulatory rules to drive the synthesis system. Unfortunately, information is sparse as to the details of the analysis system they used and as to their establishment of the LPC/tube equivalence;

- Chan and Leung [1991] have proposed to reduce the number of non-zero coefficients in both the transverse form and the lattice form of the all-pole filters, in order to provide vocoding at a lower bit-rate. Their work is developed in a signal processing framework, which does not focus on a possible physical interpretation in terms of equivalent speech production models;

- Välimäki [1995] has proposed to use unequal-length tubes that correspond to fractional delay filters to model wind instruments. The fractional delays result from the introduction of all-pass filters in the lattice structure. His developments are made in a synthesis only framework, where no focus is put on the estimation of the parameters of the model.

In this context, providing an updated mathematical development of the LPC/lossless tube equivalence in the unequal length case, such as the development given in chapter 2, is still necessary to have a deep understanding of all the physical and mathematical mechanisms involved in the equivalence. Similarly, the analysis method and the experimental results that we propose in the present chapter will provide new insights into the question of using some unequal length models for speech analysis.

In the following, the constraint made up by the unequal lengths or unequal delays will be referred to as the Non-Uniform Topology (NUT) of the lattice filter or of the equivalent tube. In a first section, we will establish the equations allowing to estimate the parameters of the NUT filters from the observation of an analyzed signal when the topology is given in advance. Then, remarking that the deduction of the topology from the signal itself may lead to a more accurate result, we will propose a method to find the optimal tube configuration while minimizing the inverse filtering residual error. Finally, a series of experiments will help us to observe the modifications of spectral modeling performance induced by the incorporation of the NUT constraint into the classical inverse filtering framework. Results showing how the NUT filters specialize in the modeling of various signals will be given and discussed.

3.1 Estimation of the reflection coefficients for a given topology

In the classical lattice formulation [Makhoul, 1977; Friedlander, 1982], the reflection coefficients usually derive from the minimization of an error criterion based on a norm of the forward residual ϵ_t^+, the backward residual ϵ_t^- or a combination of both. The norm can be the variance of the forward/backward signal, or equivalently the mean square of those signals.

In the following, we will study the modifications that the NUT constraint brings to some classical estimators. As a preliminary illustrative example, the complete development of an estimator will be given in the case of Burg's method [Burg, 1978].

3.1.1 Detailed development of a NUT estimator through Burg's method

Definition of a Least Mean Square error criterion

Burg's method is based on the analytic minimization of a modeling error criterion which corresponds to the sum of the mean squared backward and forward residual errors :

$$\xi^2(p) = \frac{1}{2}\left\{ \sum_{t=p+1}^{N} \left[\epsilon_t^+(p)\right]^2 + \sum_{t=p+1}^{N} \left[\epsilon_t^-(p)\right]^2 \right\} \tag{3.1}$$

At each step (p), the mean-square of the forward and backward errors is computed over a window of length $N-p$, where p represents the number of unit delays met at the previous steps of the lattice.

In the case of NUT lattices, the total delay is shifted due to the potential presence of some non-unit delays. Let Σ_p denote the sum of all the delays n_k from order 0 to order p:

$$\Sigma_p = \sum_{k=0}^{p} n_k \tag{3.2}$$

Introducing this sum in the above criterion, and replacing the index p by $p+1$ to simplify some upcoming expressions, we get :

$$\xi^2(p+1) = \frac{1}{2}\left\{ \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^+(p+1)\right]^2 + \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^-(p+1)\right]^2 \right\} \tag{3.3}$$

The figure 3.1 illustrates the principle of this criterion.

Introducing the parameter k_{p+1} in view of the analytic minimization of $\xi^2(p+1)$, we obtain:

$$\xi^2(p+1) = \frac{1}{2}\left\{ \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^+(p) + k_{p+1}\epsilon_{t-n_p}^-(p)\right]^2 + \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_{t-n_p}^-(p) + k_{p+1}\epsilon_t^+(p)\right]^2 \right\} \tag{3.4}$$

where n_p is the delay of the p^{th} cell.

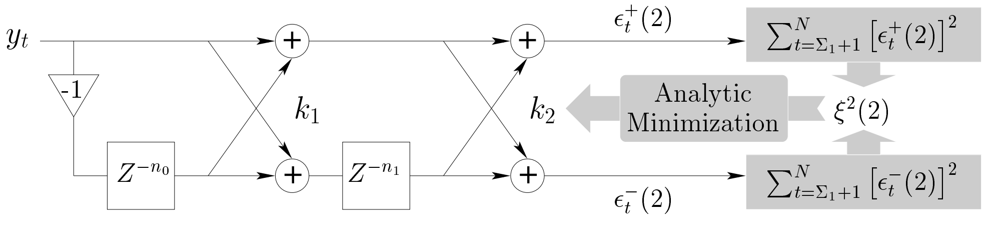


Figure 3.1: Burg's error criterion.

Minimization of the Mean Squared Error criterion

In the case of the classical Burg's method, an optimal estimator for the coefficient k_{p+1} is found at each stage $(p+1)$ of the lattice by the analytic minimization of the mean squared error $\xi^2(p+1)$. In the case of the unequal delays, performing the analytic minimization is still possible:

$$\begin{aligned} \frac{\partial \xi^2(p+1)}{\partial k_{p+1}} &= 0 \qquad (3.5) \\ &= 2\sum_{t=\Sigma_p+1}^{N} \epsilon_t^+(p)\epsilon_{t-n_p}^-(p) + k_{p+1}\left\{\sum_{t=\Sigma_p+1}^{N}\left[\epsilon_t^+(p)\right]^2 + \sum_{t=\Sigma_p+1}^{N}\left[\epsilon_{t-n_p}^-(p)\right]^2\right\} \end{aligned}$$

$$\Rightarrow \boxed{k_{p+1}^B = \frac{-2\sum_{t=\Sigma_p+1}^{N} \epsilon_t^+(p)\ \epsilon_{t-n_p}^-(p)}{\sum_{t=\Sigma_p+1}^{N}\left[\epsilon_t^+(p)\right]^2 + \sum_{t=\Sigma_p+1}^{N}\left[\epsilon_{t-n_p}^-(p)\right]^2}} \qquad (3.6)$$

The equation (3.6) defines an optimal estimator for the coefficient k_{p+1} at each step $(p+1)$ of the lattice growth, given a particular lattice topology (or, equivalently, given a particular repartition of delays). The application of this estimator is illustrated in the figure 3.2.

More generally, since this procedure is optimal for each step $(p+1)$ with respect to the minimization of $\xi^2(p+1)$, the recursive application of the estimator provides a globally optimal solution $\{k_i;\ i = 1 \cdots M\}$ when the whole topology is specified (i.e., when the respective delays of the M cells are fixed in advance).

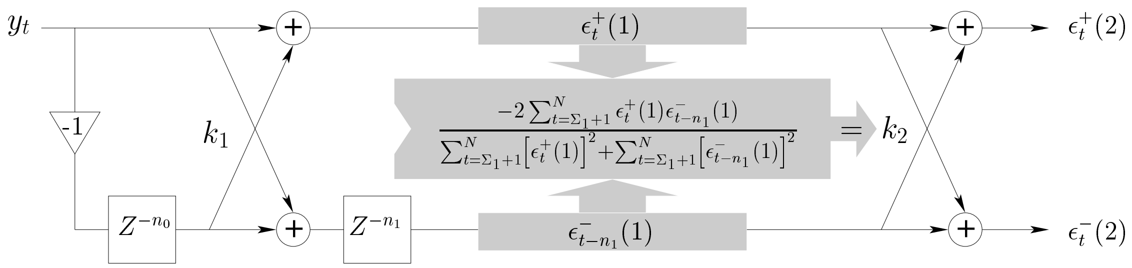


Figure 3.2: Application of Burg's estimator.

3.1.2 Other possible estimators

Other possible estimators are given in table 3.1. The forward estimator k^f_{p+1} (resp. the backward estimator k^b_{p+1}) corresponds to the analytic minimization of the mean square of ϵ^+_t (resp. ϵ^-_t) over the same window span as used for the Burg error criterion. From (3.6) and the linearity of the differentiation, it is clear that k^B_{p+1} is the harmonic mean of k^f_{p+1} and k^b_{p+1}.

Alternate ways of combining the forward and backward estimators lead to :

- the minimum method, where the estimator k^f_{p+1} or k^b_{p+1} having the least absolute value is chosen for each cell;
- the Itakura-Saito estimator, which was originally based on a stochastic formulation [Itakura and Saito, 1972] but can be interpreted as the geometric mean of k^f_{p+1} and k^b_{p+1};
- the generalized mean method, which considers the r^{th} mean of k^f_{p+1} and k^b_{p+1}.

These NUT estimators differ from their classical unconstrained counterparts only in the lags and boundaries used to measure the autocorrelations and intercorrelations of ϵ^+_t and $\epsilon^-_{t-n_p}$.

3.1.3 Interpretations of the NUT constraint in the analysis framework

Partial correlation measures

It can be demonstrated [Itakura and Saito, 1972; Friedlander, 1982] that the reflection coefficients relate to some measure of the partial correlations which characterize the analyzed signal y_t. As a matter of fact, in the stochastic formulation of spectral analysis, the spectral density of y_t is the Fourier transform of a set of τ autocorrelation coefficients :

$$r_\tau = \langle y_t \ , \ y_{t-\tau} \rangle \tag{3.7}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product. The autocorrelation coefficients r_τ can be viewed as a measure of the linear dependency between y_t and $y_{t-\tau}$. They can be gathered in a correlation matrix to solve directly for the prediction coefficients a_i, as expressed by the Yule-Walker equations [Friedlander, 1982].

But since y_t results from a prediction process, the elements of the vector $\{y_t;\ t = 1, \cdots, N\}$ are not independent. Hence, the coefficients r_τ represent a multi-linear dependency with some redundancy, and each r_τ depends in turn on $\{r_{\tau'};\ \tau' \neq \tau\}$.

Method	Development paradigm	Estimator for k_{p+1}
Forward	Minimization of: $\xi^2(p+1) = \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^+(p+1)\right]^2$	$k_{p+1}^{\boldsymbol{f}} = \frac{-\sum_{t=\Sigma_p+1}^{N} \epsilon_t^+(p)\,\epsilon_{t-n_p}^-(p)}{\sum_{t=\Sigma_p+1}^{N}\left[\epsilon_t^+(p)\right]^2}$
Backward	Minimization of: $\xi^2(p+1) = \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^-(p+1)\right]^2$	$k_{p+1}^{\boldsymbol{b}} = \frac{-\sum_{t=\Sigma_p+1}^{N} \epsilon_t^+(p)\,\epsilon_{t-n_p}^-(p)}{\sum_{t=\Sigma_p+1}^{N}\left[\epsilon_{t-n_p}^-(p)\right]^2}$
Minimum	Minimum absolute value of k_{p+1}^f and k_{p+1}^b.	$k_{p+1}^{\boldsymbol{M}} = S \cdot \min\left(\lvert k_{p+1}^f\rvert, \lvert k_{p+1}^b\rvert\right)$
Burg	Min$^{\underline{o}}$ of: $\xi^2(p+1) = \frac{1}{2}\left\{\sum_{t=\Sigma_p+1}^{N}\left[\epsilon_t^+(p+1)\right]^2 + \sum_{t=\Sigma_p+1}^{N}\left[\epsilon_t^-(p+1)\right]^2\right\}$	$k_{p+1}^{\boldsymbol{B}} = \frac{-2\sum_{t=\Sigma_p+1}^{N} \epsilon_t^+(p)\,\epsilon_{t-n_p}^-(p)}{\sum_{t=\Sigma_p+1}^{N}\left[\epsilon_t^+(p)\right]^2 + \sum_{t=\Sigma_p+1}^{N}\left[\epsilon_{t-n_p}^-(p)\right]^2}$
Itakura-Saito	Geometric mean of k_{p+1}^f and k_{p+1}^b.	$k_{p+1}^{\boldsymbol{IS}} = \frac{-\sum_{t=\Sigma_p+1}^{N} \epsilon_t^+(p)\,\epsilon_{t-n_p}^-(p)}{\sqrt{\sum_{t=\Sigma_p+1}^{N}\left[\epsilon_t^+(p)\right]^2 \cdot \sum_{t=\Sigma_p+1}^{N}\left[\epsilon_{t-n_p}^-(p)\right]^2}}$
General Mean	Generalized r^{th} mean of k_{p+1}^f and k_{p+1}^b.	$k_{p+1}^{\boldsymbol{r}} = S \cdot \left[\frac{1}{2}\left(\lvert k_{p+1}^f\rvert^r + \lvert k_{p+1}^b\rvert^r\right)\right]^{1/r}$

Notes: • Σ_p denotes the sum of all the delays n_k from order 0 to order p: $\Sigma_p = \sum_{k=0}^{p} n_k$.

• $S = \text{sign}(k_{p+1}^f) = \text{sign}(k_{p+1}^b)$. (It can be demonstrated that k_{p+1}^f and k_{p+1}^b always have the same sign [Makhoul, 1977].)

Table 3.1: Various estimators for the reflection coefficients of the inverse NUT lattice filters.

Alternately, the notion of partial correlation defines a set of coefficients that get rid of this redundancy. If $\epsilon_t^+(\tau)$ (resp. $\epsilon_{t-1}^-(\tau)$) denotes the residual of the forward (resp. backward) inverse linear filtering of y_t for the order τ, the partial correlation coefficients [Anderson, 1971] are defined recursively by :

$$k_{\tau+1} = \frac{\left\langle \epsilon_t^+(\tau)\ ,\ \epsilon_{t-1}^-(\tau) \right\rangle}{\sqrt{\left\langle \left(\epsilon_t^+(\tau)\right)^2\ ,\ \left(\epsilon_t^+(\tau)\right)^2 \right\rangle \left\langle \left(\epsilon_{t-1}^-(\tau)\right)^2\ ,\ \left(\epsilon_{t-1}^-(\tau)\right)^2 \right\rangle}} \tag{3.8}$$

In this definition, the 2-norm is applied to compute the autocorrelations. In this case, when the scalar products are developed over a given observation window, (3.8) corresponds to the Itakura-Saito estimator. Similarly, if the r-norm is applied instead of the 2-norm, the partial correlation measurement will correspond to the generalized r^{th} mean estimators. Burg's estimator also relates to this expression, since the harmonic mean can be interpreted as an approximation[1] of the geometric mean operated in the Itakura-Saito estimator [Makhoul, 1977].

Getting back to the interpretation of the NUT constraint, it is now clear that this constraint allows to control more precisely the measurement of the linear dependencies which build the signal y_t. Partial correlations are measured in a selective manner: the lags between ϵ_t^+ and $\epsilon_{t-n_p}^-$ are ruled by the non-unit delays n_p, and some of the possible lags between 0 and Σ_p are disregarded because they correspond to a combination of delays that is not authorized by the production model.

The covariance-lattice formulation

To clarify a bit more the interpretation of the constraint setting a particular k_p to zero, it may be useful to compare it with a parameter quantization problem where no bits are allocated to k_p.

It is common to treat the problems involving quantized reflection coefficients with a lattice filter formulation, because this formulation avoids the use of some optimality assumptions in the recursive computation of the parameters [Makhoul, 1977]. As a matter of fact, the covariance measures which define the reflection coefficients can be expressed in terms of some linear combinations of the signal covariance and of the linear prediction coefficients [Makhoul, 1977] :

$$F_t(p) = E\left\{\left[\epsilon_t^+(p)\right]^2\right\} = \sum_{k=0}^{p}\sum_{i=0}^{p} a_k^{(p)} a_i^{(p)} \phi(k,i) \tag{3.9}$$

$$B_t(p) = E\left\{\left[\epsilon_t^-(p)\right]^2\right\} = \sum_{k=0}^{p}\sum_{i=0}^{p} a_k^{(p)} a_i^{(p)} \phi(p+1-k, p+1-i) \tag{3.10}$$

$$C_t(p) = E\left\{\epsilon_t^+(p)\,\epsilon_{t-1}^-(p)\right\} = \sum_{k=0}^{p}\sum_{i=0}^{p} a_k^{(p)} a_i^{(p)} \phi(k, p+1-i) \tag{3.11}$$

[1] This approximation reduces the computational load by avoiding to execute the square root.

where :

- $F_t(p)$ is the covariance of $\epsilon_t^+(p)$, $B_t(p)$ the covariance of $\epsilon_t^-(p)$, $C_t(p)$ the inter-covariance of $\epsilon_t^+(p)$ and $\epsilon_{t-1}^-(p)$;
- t is the discrete time; $\epsilon_t^+(p)$ (resp. $\epsilon_t^-(p)$) is the forward (resp. backward) residual signal after the p^{th} cell; $E\{.\}$ denotes the expectation operator;
- $a_i^{(p)}$ are the prediction coefficients at step (p), and are related to the reflection coefficients by :

$$\begin{cases} a_0^{(p)} = 1 \\ a_j^{(p)} = a_j^{(p-1)} + k_p a_{p-j}^{(p-1)}; \quad 1 \leq j \leq p-1 \\ a_p^{(p)} = k_p \end{cases} \tag{3.12}$$

- $\phi_t(k,i) = E\{y_{t-k}\, y_{t-i}\}$ is the covariance of the analyzed signal y_t (and the subscript t of ϕ_t is dropped to clarify the formulae);

In the stationary case, the covariance of the analyzed signal reduces to the autocorrelation :

$$\phi(k,i) = R(i-k) = R(k-i) \tag{3.13}$$

In the case of the classical Levinson algorithm [Rabiner and Juang, 1993], which does not use a lattice formulation, the reflection coefficients are defined as [Makhoul, 1977] :

$$k_{p+1} = \frac{-\sum_{k=0}^{p} a_k^{(p)}\, R(p+1-k)}{(1-k_p^2)\, F_t(p-1)} \tag{3.14}$$

This formulation assumes the strict optimality of k_p, since the denominator of (3.14) corresponds to computing the correlation of the forward error in a recursive manner : errors made on the successive k_i propagate and end up with modifying significantly the measure $F_t(i)$. By extension, these errors corrupt the analysis of the correlation of the input signal itself.

Conversely, in the lattice case, it is possible to define the reflection coefficients as :

$$\begin{aligned} k_{p+1} &= \frac{-2\, C_t(p)}{F_t(p) + B_{t-1}(p)} \\ &= \frac{-2 \sum_{k=0}^{p} \sum_{i=0}^{p} a_k^{(p)}\, a_i^{(p)}\, R(p+1-i-k)}{\sum_{k=0}^{p} \sum_{i=0}^{p} a_k^{(p)}\, a_i^{(p)}\, R(i-k) + \sum_{k=0}^{p} \sum_{i=0}^{p} a_k^{(p)}\, a_i^{(p)}\, R(k-i)} \end{aligned} \tag{3.15}$$

In this formulation[2], the true correlation $R(i,k)$ is used at each step (p). Hence, if an error (or a "modification") is made on the value of k_p, it is transmitted to the subsequent values of $a_i^{(p)}$ after the application of (3.12). The value of k_{p+1} is modified in the next step, *but the original measure of correlation of the input signal remains untouched.*

The above formulation is known as the covariance-lattice or correlation-lattice formulation, and is due to Makhoul [1977]. It can be shown that it is strictly equivalent to the application of the classical direct lattice estimators (of the type of table 3.1 in the equal-length case), but that it is computationally more efficient[3].

Coming back to the interpretation of the NUT constraint, and interpreting it with the help of the covariance-lattice formulation, we see that :

- imposing the NUT constraint in the framework of Levinson's recursion violates a necessary optimality assumption (this explains the contradiction observed in [Krstulović, 1997]);

- in the lattice framework, setting $k_{p+1} = 0$ amounts to having :

$$\sum_{k=0}^{p} \sum_{i=0}^{p} a_k^{(p)} a_i^{(p)} R(p+1-i-k) = 0 \tag{3.16}$$

  which represents a constraint on the values of the prediction coefficients $a_i^{(p)}$, but does not corrupt the observation of the correlation of the analyzed signal.

The lattice analysis framework, equivalent to the recursive application of (3.15) and (3.12), authorizes to constrain some reflection coefficients k_i to adopt an arbitrary constant value (e.g., 0) without betraying the nature of the analyzed signal. The true effect of the NUT estimators is therefore to re-distribute the correlation information (or covariance information) across the free reflection coefficients, according to the topological constraints imposed by the production model.

[2] (3.15) corresponds to the Burg method in the stationary case, but alternate formulations can be defined in the same manner for the forward, backward, Itakura-Saito and r^{th} mean methods, as well as for the non-stationary case where covariances do not reduce to correlations.

[3] This formulation is more efficient in terms of the number of elementary mathematical operations, but from our programming experience, it involves more cost in terms of pointer arithmetics, and therefore yields slower software implementations.

3.1.4 Stability issues

We have shown in section 2.2.4 that the transfer functions corresponding to synthesis-oriented acoustic tubes had always an all-pole form. This form describes Infinite Impulse Response (IIR) filters, which do not have a guaranteed stability.

In the lattice formulation, the stability condition translates in a very simple condition on the values of the reflection coefficients :

$$\forall p, \quad |k_p| < 1 \tag{3.17}$$

As a matter of fact, having $|k_{p+1}| < 1$ for a new cell at step $(p+1)$ guarantees that the global energy of the backward and forward signal will decrease from step (p) to step $(p+1)$. Let us verify this fact in the case of the NUT Burg estimator.

From the lattice form, we have :

$$\left\{ \begin{array}{lclcl} \epsilon_t^+(p+1) & = & \epsilon_t^+(p) & + & k_{p+1}\,\epsilon_{t-n_p}^-(p) \\ \epsilon_t^-(p+1) & = & \epsilon_{t-n_p}^-(p) & + & k_{p+1}\,\epsilon_t^+(p) \end{array} \right. \tag{3.18}$$

Hence :

$$\begin{aligned} \xi^2(p+1) &= \frac{1}{2}\left\{ \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^+(p+1)\right]^2 + \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^-(p+1)\right]^2 \right\} \qquad (3.19) \\ &= \frac{1}{2}\left\{ \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^+(p) + k_{p+1}\epsilon_{t-n_p}^-(p)\right]^2 + \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_{t-n_p}^-(p) + k_{p+1}\epsilon_t^+(p)\right]^2 \right\} \\ &= \frac{1}{2}\left\{ \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^+(p)^2 + 2k_{p+1}\epsilon_t^+(p)\epsilon_{t-n_p}^-(p) + k_{p+1}^2\epsilon_{t-n_p}^-(p)^2\right] \right. \\ &\qquad \left. + \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_{t-n_p}^-(p)^2 + 2k_{p+1}\epsilon_{t-n_p}^-(p)\epsilon_t^+(p) + k_{p+1}^2\epsilon_t^+(p)^2\right] \right\} \\ &= \xi(p)^2 + \frac{1}{2}\left\{ 4\,k_{p+1} \sum_{t=\Sigma_p+1}^{N} \epsilon_t^+(p)\epsilon_{t-n_p}^-(p) \right. \\ &\qquad \left. + k_{p+1}^2 \left(\sum_{t=\Sigma_p+1}^{N} \left[\epsilon_t^+(p)\right]^2 + \sum_{t=\Sigma_p+1}^{N} \left[\epsilon_{t-n_p}^-(p)\right]^2 \right) \right\} \end{aligned}$$

By application of the estimator (3.6) on one of the k_{p+1} in the second member of the bracketed addition, we obtain:

$$
\begin{aligned}
\xi^2(p+1) &= \xi(p)^2 \\
&\quad +2k_{p+1}\sum_{t=\Sigma_p+1}^{N} \epsilon_t^+(p)\epsilon_{t-n_p}^-(p) \\
&\quad +k_{p+1}\frac{-\sum_{t=\Sigma_p+1}^{N}\epsilon_t^+(p)\epsilon_{t-n_p}^-(p)}{\sum_{t=\Sigma_p+1}^{N}\left[\epsilon_t^+(p)\right]^2+\sum_{t=\Sigma_p+1}^{N}\left[\epsilon_{t-n_p}^-(p)\right]^2}\sum_{t=\Sigma_p+1}^{N}\left(\left[\epsilon_t^+(p)\right]^2+\left[\epsilon_{t-n_p}^-(p)\right]^2\right) \\
&= \xi(p)^2 - k_{p+1}\sum_{t=\Sigma_p+1}^{N} \epsilon_t^+(p)\epsilon_{t-n_p}^-(p) \\
&= \xi(p)^2 - k_{p+1}\left(k_{p+1}\,\xi(p)\right) \qquad (3.20)
\end{aligned}
$$

$$\Rightarrow \boxed{\xi^2(p+1) = (1-k_{p+1}^2)\xi^2(p)} \qquad (3.21)$$

Hence, for each cell added at step $(p+1)$ with a reflection coefficient $|k_{p+1}| < 1$, and following an arbitrary repartition of delays $n_{i,i<p+1}$, the MSE is guaranteed to decrease.

Of all the estimators given in table 3.1, all but the forward and backward estimators guarantee that the reflection coefficients have an absolute value below 1, and that the resulting filter will always be stable[4]. In those cases, the introduction of the NUT constraint does not disrupt the stability of the filter. The same conclusion can be more simply drawn from the fact that the NUT constraint corresponds to imposing $|k_i| = 0$ for some i, which obviously respects the condition (3.17).

It is also interesting to remark that the reflection coefficients are defined in the physical domain as:

$$k_{i+1} = \frac{S_{i+1}-S_i}{S_{i+1}+S_i} \quad \Leftrightarrow \quad \frac{S_i}{S_{i+1}} = \frac{1-k_{i+1}}{1+k_{i+1}} \qquad (3.22)$$

Hence, the stability condition corresponds to keeping positive areas for each section of the equivalent acoustic tube. The values $g_{i+1} = \frac{1-k_{i+1}}{1+k_{i+1}}$ are called Area Ratios, and are used as speech features in some speech processing applications [Viswanathan and Makhoul, 1975; Rabiner and Juang, 1993].

[4] For instance, in the Burg case:

$$
\begin{aligned}
\left|\frac{-2\sum a\,b}{\sum a^2+\sum b^2}\right| < 1 \;\Leftrightarrow\; |-2ab| &< a^2+b^2 \\
\pm 2ab &< a^2+b^2 \\
0 &< a^2 \mp 2ab + b^2 \\
0 &< (a\pm b)^2 \quad \text{QED}
\end{aligned}
$$

3.2 Optimization of the filter topology

In the section 3.1, we have developed some new estimators for the reflection coefficients when the topology of the tube or of the NUT filter is known in advance (figure 3.3; e.g., when accounting for the topology of the DRM model [Krstulović, 1998, 1999]). But various constraints, or equivalently various shifts in the measures of the signal's correlations, are likely to lead to different modeling performances. It may therefore be more accurate to try and optimize the topology than to impose a single a-priori NUT configuration. In the following, we propose to implement the search for a best topology as an additional optimization layer [Krstulović and Bimbot, 2000, 2001].

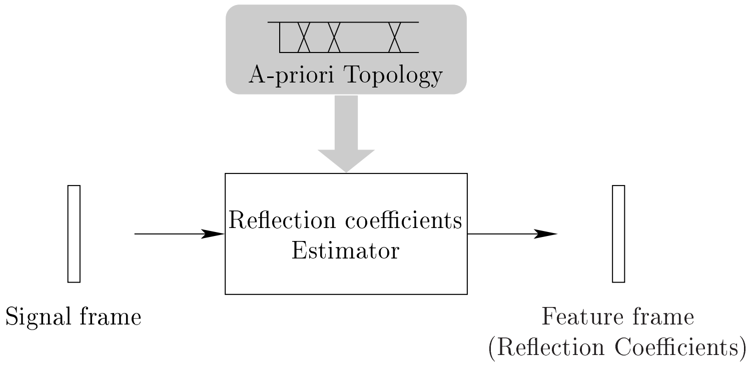


Figure 3.3: NUT feature extraction with an a-priori topology.

3.2.1 Search for the best topology

Method

The specification of a number N_s of unequal-length sections to be distributed over a given number N_u of unit sections is equivalent to the consideration of $(N_s - 1)$ free interfaces out of $(N_u - 1)$ possible positions. Hence, a $[N_s/N_u]$ couple defines a set of $\binom{N_u-1}{N_s-1}$ possible combinations[5] of delays.

To search for the best configuration in the specified set, all the lattice topologies respecting the $[N_s/N_u]$ specification can be generated and systematically used to inverse-filter some training data. The one bringing the least Mean Squared Error (MSE) over the considered training set will be regarded as the best topology.

[5]The notation $\binom{n}{p}$, sometimes noted C_n^p, corresponds to the notion of combination used in the standard combinatorics theory. It counts the number of ways of combining n objects p by p, and is mathematically defined by $\binom{n}{p} = \mathrm{C}_n^p = \frac{n!}{p!(n-p)!}$, where ! is the factorial operation.

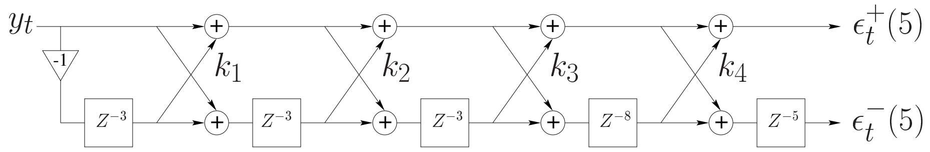
The dependency of the optimal topology to the training set corresponds to a specialization of the inverse filter to the data of interest. The training set can be defined in several ways :

1. it can correspond to a single frame of signal. In this case, the topology optimization layer finds the best topology for each signal frame (figure 3.4);
2. it can correspond to a particular speech class, e.g. instances of a particular vowel or utterances from a particular speaker. In this case, a class-specific topology will be found (figure 3.5);
3. it can consist in a greater volume of speech, to examine if the resulting speech-specific NUT topology can tell us something about the global properties of speech production by lossless tubes.

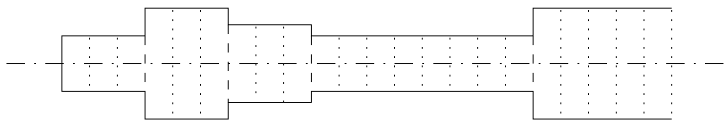
Results pertaining to the application of these different modes of topology optimization will be given in the upcoming sections.

Notation

In the following, the various NUT lattice configurations will be identified by strings starting with the number of delay blocks N_s, expressed over the number of spanned unit delays N_u, and followed by the enumeration of their order. An example would be : [5/22:3x3,8,5.], which reads : "a NUT lattice with 5 cells spanning 22 unit-delays, and which has three 3^{rd} order delays, one 8^{th} order delay and one 5^{th} order delay." The corresponding flow chart looks like :



This model is equivalent to a lossless acoustic tube made of $N_s = 5$ unequal-length sections distributed over $N_u = 22$ unit sections $(3 \times 3 + 8 + 5 = 22)$:



This notation also relates to :

- the number of degrees of freedom (DoFs), or number of free reflection coefficients. Since this number is equal to the number of interfaces delimiting the unequal-length sections, it corresponds to $(N_s - 1)$;

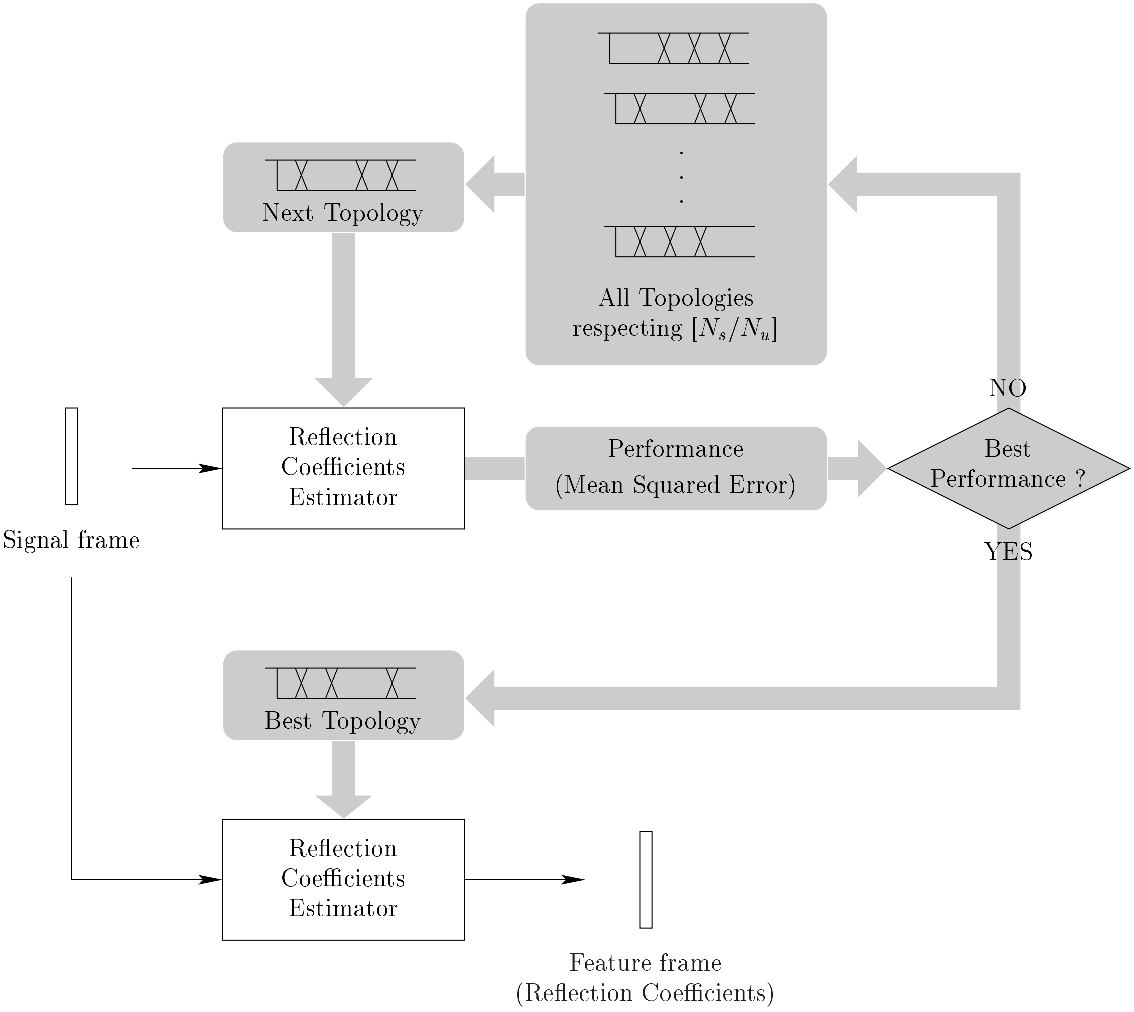


Figure 3.4: NUT feature extraction with a frame-specific topology optimization.

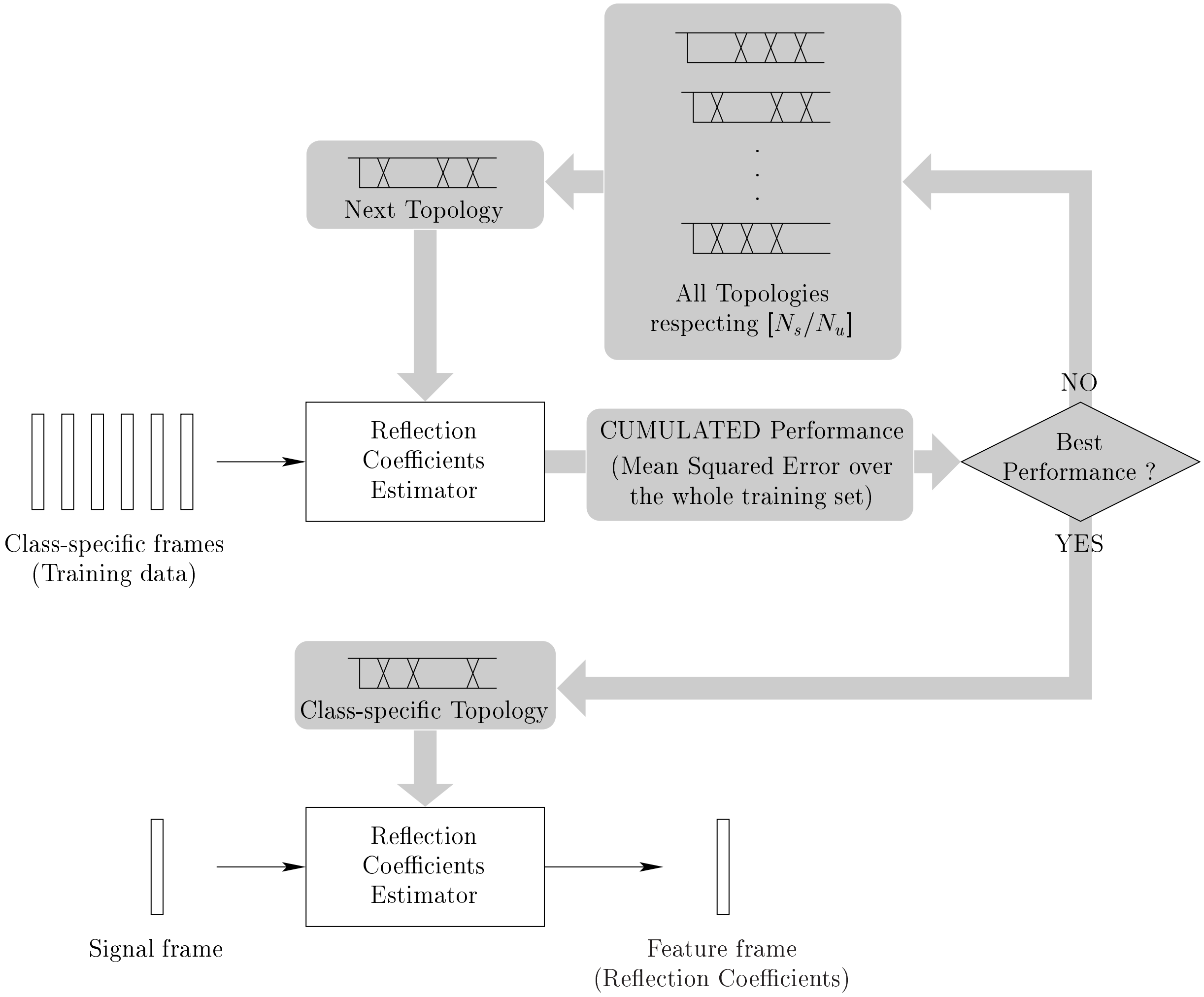


Figure 3.5: NUT feature extraction with a class-specific topology optimization.

- the global LPC order of the NUT lattice, which is equal to the sum of the delays' order minus the order of the last delay (or sum of the sections' lengths minus the length of the last section).

In our example, the [5/22:3x3,8,5.] model corresponds to a lattice with 4 DoFs (4 unconstrained reflection coefficients), but it leads to a 17^{th} order all-pole model ($22 - 5 = 17$).

3.2.2 Computational considerations

Base algorithm

The enumeration of a $[N_s/N_u]$ set of topologies corresponds to a tree-structured process (fig. 3.6), where:

- each new level corresponds to the addition of a new degree of freedom (a new mobile interface);
- the tree has N_s levels;
- the values at the nodes identify the order of the newly added delay, or equivalently the length of the newly added tube section;
- the valid branches correspond to the paths where the node values sum up to the global number N_u of unit delays.

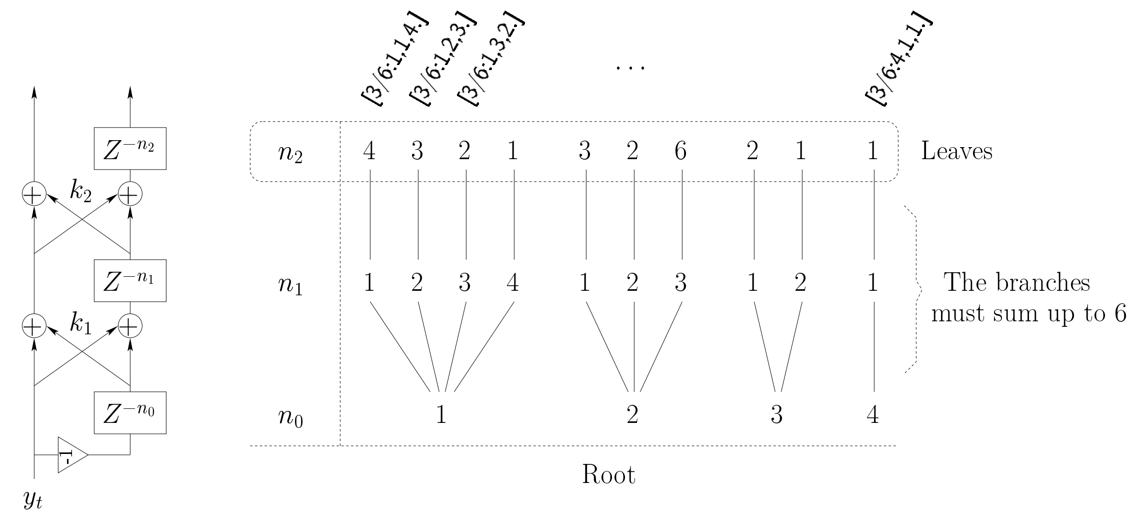


Figure 3.6: The tree corresponding to a [3/6] specification.

The global computational load depends both on the number of tested filters, imposed by the $[N_s/N_u]$ specification, and on the length of the observation window used for the estimation of the correlations. The complexity of the algorithm is of the order of:

$$N_w \cdot (N_s - 1) \cdot e^{(N_u - 1)} \tag{3.23}$$

where N_w is the number of samples in the observation window, $(N_s - 1)$ is the number of free reflection coefficients and $(N_u - 1)$ is the total number of available positions. The exponential term derives from the enumeration of the $\binom{N_u-1}{N_s-1}$ possible filters.

However, testing each of the resulting filters individually implies some redundancy: the values of the reflection coefficients for lower indexes are not modified if the delays at higher indexes are varied. This redundancy can be eliminated by organizing the estimation of the reflection coefficients and of the residual errors in accordance with the depth-first browsing of the tree: the results obtained for the lower levels can be re-used when following a particular branch upwards. The complexity is then reduced to a value lying between $N_w \cdot (N_s - 1) \cdot e^{(N_u - 1)}$ and $N_w \cdot (N_s - 1) + e^{(N_u - 1)}$. In any case, it remains exponential with the number of unit sections N_u[6]. Note that if one wants to preserve the consistency between the filters and the tube models, N_u is in turn related to the sampling frequency of the analyzed signal.

A straightforward but naive way of realizing the depth-first browsing of the tree consists in using a recursive function. This solution leads to a readable implementation but entails a significant memory and function calls overhead. Instead, we have implemented a more efficient but less straightforward solution which uses a single loop. As an example of the order of magnitude of the "real world" computation time, the exhaustive search of 116'280 filters, for a [8/22] specification with an observation window of 556 samples and using the NUT-Burg estimator, takes an average time of 1.8 seconds on a PentiumIII@1GHz. For a frame rate of 10ms, this corresponds to about 200 times real time.

To make the number of tested filters more tractable, further constraints can be imposed to the optimization scheme, such as a minimum delay order. For instance, in the case of the distribution of 7 degrees of freedom (delimiting 8 sections) on a 31^{st} order LPC process (32 unit sections), 2'629'575 filters have to be tested. Imposing the minimum delay to be no shorter than 2 units reduces this number to 245'157 filters.

[6] Chan and Leung [1991] propose to use a dynamic programming algorithm to optimize the values of the unequal delays in a "sparse" lattice filter similar to the model studied here. They claim a complexity of the order of $N_s \cdot N_u^2$. However, we believe that their method constrains the position of the free reflection coefficients in a sequential way which may not necessarily correspond to an optimal solution. We did not have a chance to experiment their solution and to compare it to ours before the completion of the present thesis.

Potential refinements

We have also investigated some ways of pruning the search tree by exploiting the dependency of the filtering error on the levels of the tree. For instance, we have tried to determine some relative upper bounds for the filtering performance of the branches starting at a given node. This would have enabled us to sort and prune these branches without browsing them up to the leaves level. In a first attempt, we have tried to set all the delays following a particular node to 1, up to the global model order and without imposing further NUT constraints in the higher levels of the lattice. Experiments proved this approach to be wrong : on a long test sentence, 19% of the frame-dependent topologies found with this pruning scheme were different from the solution issued by an exhaustive search.

As a matter of fact, it seems that no a priori ranking can be determined between the performances of filters with a different number of degrees of freedom. Because of the various layouts for the correlation measures, the number of DoFs for *non-optimal* topologies is no more an absolute indicator of the filtering performance : filters with less DoFs may perform better if their topology happens to be more closely related to the structure of the analyzed signal. Alternately, when probing the lower levels, trying to keep a fixed number of DoFs while respecting the specified global order falls back on the exhaustive search.

Globally, the decoupling of the number of DoFs from the LPC order entails a dependency of the filtering performance on the whole filter structure, rather than on the step-by-step accurate estimation of the parameters of each individual cell. The values of each reflection coefficient at each lattice stage become dependent on the whole filter topology. The topology represents a "longitudinal" constraint, which ties the respective dynamic ranges of the reflection coefficients, and hence determines globally the filter performance for a particular signal. Hence, to our current knowledge, only the comparison of *optimal* configurations makes sense when ranking the topologies that have an unequal number of degrees of freedom.

Practical issues

In the absence of a more detailed understanding of the explicit relationship between the filter performance, the form of the topology and the dependency upon the analyzed signal, we have preferred to try and find a practical solution to the speed-up of the exhaustive search, to be able to carry on with the experiments rather than spending all our time on the development of a pruning algorithm.

As a matter of fact, distributed computation can be used to speed up the computation. When dealing with a whole database, our approach has consisted in spreading the feature extraction process across the machines of a whole network[7], using a load control mechanism to

[7] IDIAP and EPFL's LTS lab networks, comprising SUN Ultras with 440/330/110MHz CPUs and Linux PCs

avoid disturbing the usual network activity (i.e., launching processes on the "free" CPUs only, and letting the processes of other users override our occupancy of a particular CPU). With about 40 machines of various powers available at any time on average, the feature extraction for the 1860 files composing the "Citation" tasks of the University of Wisconsin database (see appendix A) took about 30 hours for a frame-by-frame topology optimization and feature extraction (116'280 configurations per frame, corresponding to a [8/22] specification, with 25ms observation windows).

We acknowledge that the resources needed to achieve this somewhat "reasonable" computation time are more compatible with a research framework than with everyday life applications. However :

- a more efficient algorithm than plain exhaustive search may be found if a scheme for pruning the search tree without disturbing the optimality proves to be realizable;
- with no more thought effort, faith can be placed in the growth of the computational power of low-cost computers to make such heavy computational problems accessible to the applicative domain;
- our distributed feature extraction scheme used potentially wasted CPU resources, since it operated only during the time slots where target CPUs were idle; this touches another trend of modern computing, that of distributed computing across the free resources of whole networks rather than within a single machine.

After the review of these computational issues, which have cost us a significant amount of time in terms of software design and development, let us review some experiments aiming at assessing the accuracy of the NUT modeling scheme.

3.3 Experimental results

The experiments that we have realized aim at studying two different aspects of the NUT modeling method :

1. the evolution of the inverse filtering and spectral modeling accuracy in relation with the reduction of the number of degrees of freedom and in relation with the various possible filter configurations;
2. the interpretation of the information represented in the unequal-length topologies, and therefore the possibility to use the topologies as an analysis tool.

Results pertaining to these two topics will be given in the following sections.

with 1GHz and 500MHz CPUs.

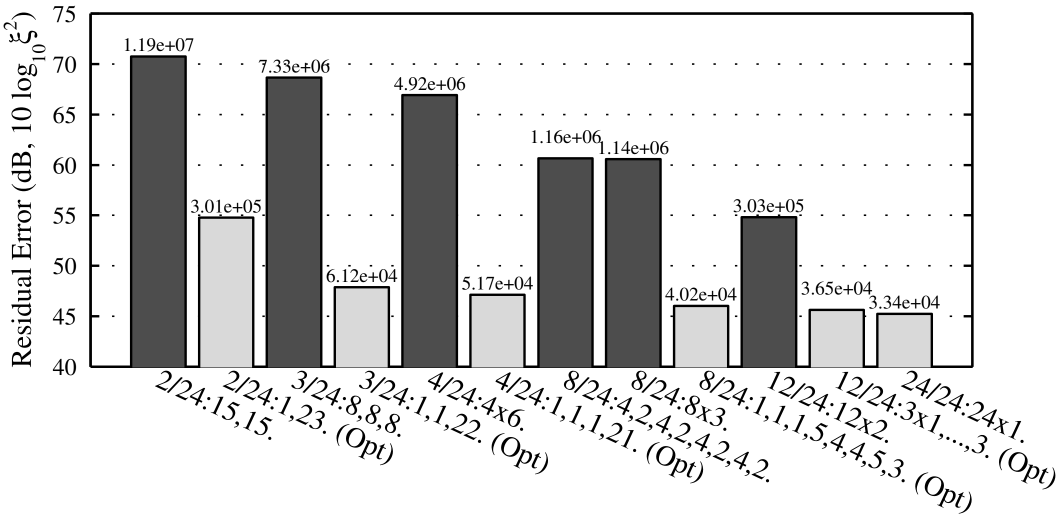


Figure 3.7: Comparison of Log Residual Errors for various NUT lattice models (for a frame of vowel /a/ @ 24kHz). The original value of ξ^2 is indicated above the bars. The light bars correspond to the optimal NUT configurations, while the dark bars correspond to some a-priori topologies.

3.3.1 Accuracy of the optimized NUT filters

Inverse filtering accuracy

Figure 3.7 shows the value of the relative mean squared residual error (MSE) after the application of variously constrained filters on a frame of vowel /a/ (taken as the whole training set, for a frame-based optimization of the topology). This frame is drawn from a test sentence spoken by a French male speaker : "*dans cette crémerie, on mange du fromage fort; je ne peux atteindre les bocaux de confiture*" (anechoic chamber, 32kHz resampled to 24kHz).

From 24 unit sections, the number of degrees of freedom (DoFs) has been constrained to the values 1, 2, 3, 7, 11 and 23 (no constraint). On the figure, for each of these DoF reductions, an a-priori configuration (dark bars; two a-priori configurations in the case of 7 DoFs) is compared with the configuration found by the optimization algorithm (light bars).

If we focus on the comparison between an a-priori configuration and an optimal one for a particular number of DoFs, the figure shows that the optimization of the topology is able to bring a reduction of the MSE for a given number of degrees of freedom. Various topologies will give errors ranging from the optimal one to at least the error shown in the case of the chosen a-priori cases. The optimization of the topology is therefore able to specialize the structure of the filter to the analysis of a given signal, given a $[N_s/N_u]$ specification.

It is important to note that the comparison made in figure 3.7 does not prejudge of the relative amplitude of the achievable MSE reduction : the large differences in performance observed in

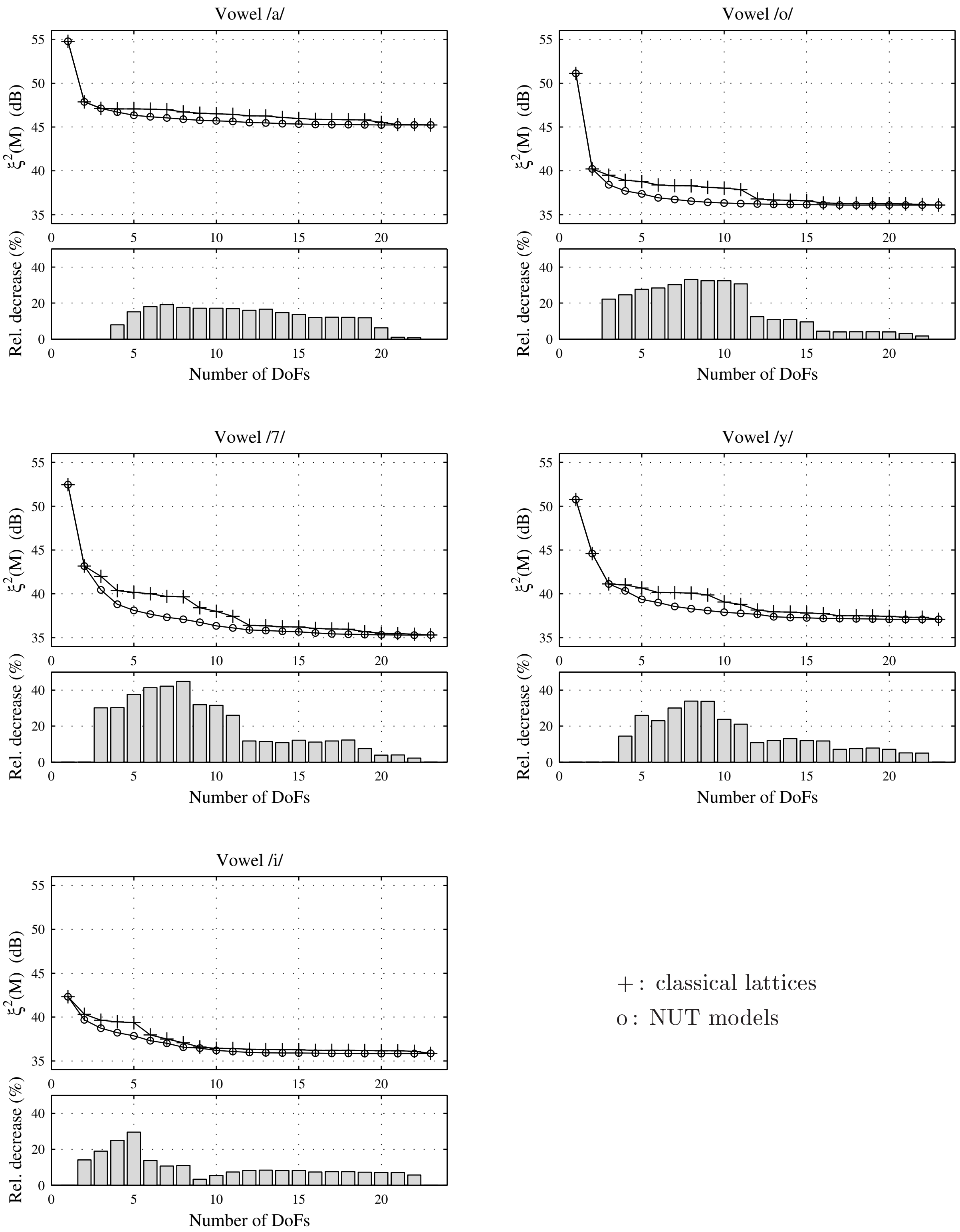


Figure 3.8: MSE decrease in function of the number of Degrees of Freedom, for various vowels.

this figure for the white and dark bars simply indicates that the employed a-priori topologies are particularly ill suited to the modeling of the analyzed test frame. The specialization, in itself, consitutes the significant phenomenon. In this respect, the optimization of the topology represents an essential step in the NUT modeling framework.

Now, focusing on the light bars only, the figure 3.7 also shows, as could be expected, that the optimal error decreases monotonically with the increase in the number of preserved degrees of freedom. Figure 3.8 gives some additional results in the case of other vowels drawn from the same test sentence[8]. There, a comparison is operated between the optimal NUT lattices of the type [n/24] and the unconstrained LPC lattices with an equivalent order. The figure shows that the optimal NUT filters *produce a lower residual error than the classical LPC with an equal number of DoFs.* The observed error reductions typically range from a few percent to about 45% in the case of the vowel /7/.

Two different phenomena play a role in these MSE variations. In the case of the unconstrained predictors, the decrease in the modeling accuracy is induced by the reduction of the prediction time span (going with the reduction of the LPC order). In the case of the constrained predictors, the decrease in MSE results from the reduction of the number of DoFs, while the prediction time span stays the same. Hence, a [8/24] predictor still uses 23 speech samples for its prediction, whereas a [8/8] predictor uses only 7 samples. Given that the two curves necessarily join at their lower end ([24/24], equivalent to the LPC order 23), it appears that the loss induced by the reduction of the time span is greater than the loss induced by the reduction of the number of DoFs.

Figure 3.9 shows the decrease of the residual error for the reverse experiment, namely keeping a fixed number of parameters and augmenting the global order of the NUT lattice. In this figure, the flat portions of the curve represent the zones where only the last delay is increased in the successive topologies found by the optimization. Changing the last delay does not change the transfer function of the forward predictor: it affects the order of the backward predictor only. The figure shows that the accuracy of the inverse filter can slightly augment with the global order, while keeping a fixed number of parameters for the model description.

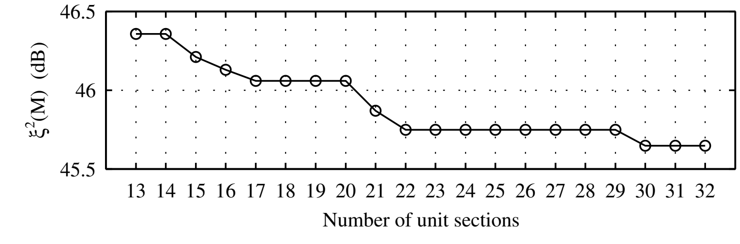


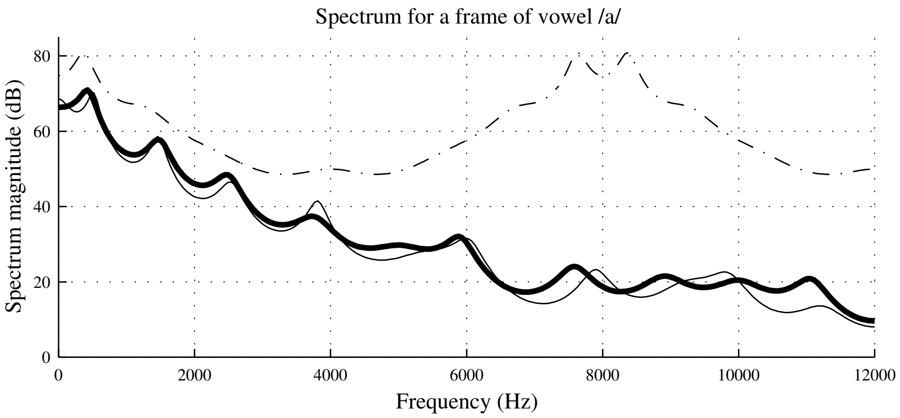
Figure 3.9: Residual error decrease versus increase in the global order, for a frame of vowel /a/ @ 24kHz, when going from [13/13] to [13/32] (12^{th} order LPC $\rightarrow$ 31^{st} order NUT).

[8]The Worldbet phonetic notation [Hieronymus, 1993; Lander, 1997] is used throughout the present report to identify the vowels.

Spectral modeling accuracy

By definition, the mean squared residual error is directly proportional to the spectral fitting accuracy of a particular inverse filter [Makhoul, 1975a,b]. Hence, the NUT filters should be able to capture spectral shapes within a certain margin of accuracy.

The figure 3.10 compares the spectral shapes obtained with Burg estimators[9] in three different cases corresponding to a 23$^{\text{rd}}$ order LPC : a classical [24/24] model, an optimized [8/24] NUT lattice and an a-priori [8/24] NUT configuration. The figure adds to the idea that the optimization of the topology is essential to the framework of NUT modeling.



continuous line (——) : 23^{rd} order LPC – [24/24:24x1.];
bold line (▬▬) : optimal NUT lattice w/ 8DoFs – [8/24:1,1,1,5,4,4,5,3.];
dash-dot ($- \cdot -$) : a-priori NUT configuration w/ 8DoFs – [8/24:8x3.].

Figure 3.10: Spectral accuracy of the optimized vs. unoptimized lattice configurations.

As a matter of fact, while the optimized NUT is able to capture the speech formants with a reasonable accuracy, the a-priori [8/24:8x3.] model is totally unadapted to the modeling of the analyzed frame. This particular a-priori topology is only able to model the spectra showing a periodicity of one third the sampling frequency since, in this case, we have :

$$H(z) = H(z^{-3}) = H(e^{-j\omega\frac{3}{F_s}})$$

which is equivalent to sampling the signal at $\frac{F_s}{3}$, and which brings periodicity (or aliasing) in the range $0 < f < F_s$. (Hence the repetition, around 8000Hz, of the peak occurring close to frequency 0 in figure 3.10.) Since the spectrum of the vowel /a/ does not feature such a

[9] From the reflection coefficients, the transfer function $H(z)$ can be deduced via the relation (2.3). The evaluation of $H(z)$ along the upper half of the unit circle, for the complex variable z, gives the smoothed spectrum [Makhoul, 1975b]. Only the spectral magnitudes are considered here.

characteristic periodicity, the modeling error of [8/24:8x3.] is necessarily huge for this frame. The same type of phenomenon will occur with all the structures where the greatest common divisor of the lengths of the sections is not 1. For instance, this is the case with the a-priori topologies which have been used in figure 3.7, and which show large modeling errors.

The figure 3.11 shows the spectral fitting performance of the optimal [8/24] NUT lattices for various other vowels. It is clear that the [8/24] NUT optimized for each frame is able to retain a good spectral modeling accuracy *while dividing the number of necessary parameters by 3*: 7 parameters can be used to drive 23 poles, given an adequate NUT configuration.

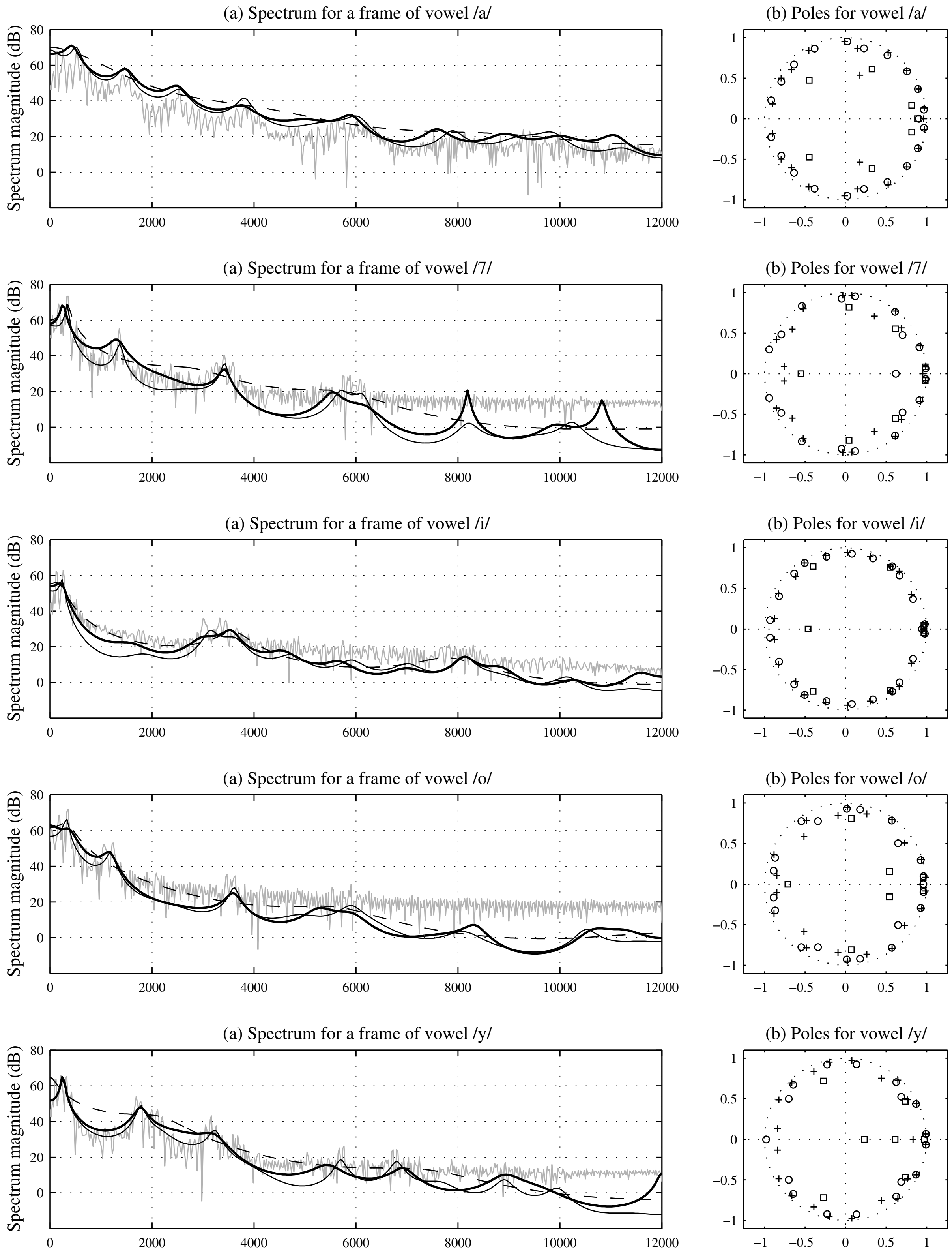
These results confirm that with NUT lattices, part of the signal coding is transfered from the coefficient values to the filter structure. A coding scheme exploiting this model would replace p conventional reflection coefficients (or log area ratios [Viswanathan and Makhoul, 1975]) with a lower number of coefficients distributed over a p^{th} global order, plus a codeword to index the optimal filter topology. The set of topologies searched during the optimization could be pruned after a statistical study on databases containing many instances of the signals of interest (e.g., speech), in order to remove the unused filter structures. By eliminating the unused or seldom used topologies from the search, the optimization would be made faster, and the number of bits necessary to index the topologies would be reduced. The quality compromise found by adjusting all these specifications, in addition to an efficient coefficient quantization method, may allow to reach a better coding quality at a lower bit rate than the unspecialized LPC models.

NUT filters appear to be a valid spectral analysis model. But further experiments are needed to determine whether the topologies reached by the second optimization layer are also useful from an analysis point of view, i.e. if they are informative about some actual acoustic phenomena.

3.3.2 Optimal configurations and dependency upon the signal

Topologies versus synthetic signals

To determine the nature of the information captured by the NUT topologies, the optimization of the topologies has been first applied to individual frames of non-speech test signals sampled at 32kHz and spanning 25 milliseconds (800 samples). With 100 different frames of Gaussian white noise, a [8/32] optimization yielded 100 different topologies showing no special pattern. Conversely, frames of sine waves with random phase values consistently gave the topologies shown on table 3.2. These results are of course independent of the signal's amplitude (various amplitudes corresponding roughly to the standard deviation of speech have nevertheless been checked).



(a) Spectra superimposed over the greyed FFT: 23^{rd} order LPC [24/24] (continuous line); optimal NUT lattice w/ 8DoFs [8/24] (bold line) ; 7^{th} order LPC [8/8] (dashed).
(b) Corresponding poles: [8/8] (□), optimal [8/24] (o) and [24/24] (+).

Figure 3.11: Constrained vs. unconstrained spectra for various vowels.

Sine wave	Best configuration
1000 Hz	[8/32: 16,1,1,1,1,1,1,10.]
2000 Hz	[8/32: 8,1,1,1,1,1,1,18.]
4000 Hz	[8/32: 4,1,1,1,1,1,1,22.]
8000 Hz	[8/32: 1,1,1,1,1,1,1,26.]

Table 3.2: Best configurations for various sine waves with a [8/32] NUT filter.

These results confirm that the topology optimization layer is able to capture the lags with maximum correlation from the signal, and to reflect them in the structure of the NUT lattice.

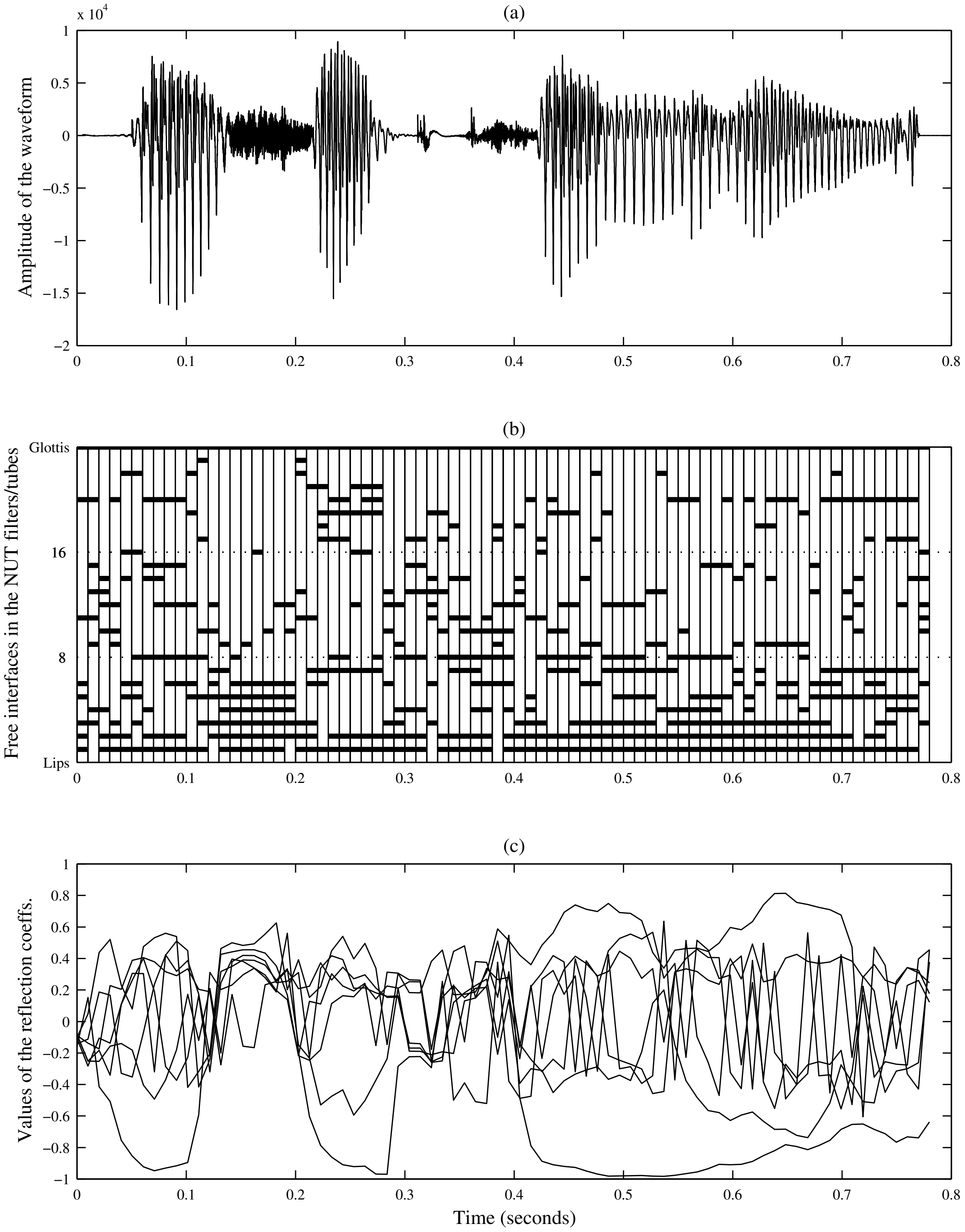
Topologies versus frames of speech signals

The figure 3.12 shows the results of a [8/24] NUT analysis applied to a a French test sentence ("*dans cette crémerie*") spoken by a male speaker, recorded in an anechoic chamber with a 32kHz sampling frequency and resampled to 24kHz prior to the NUT analysis. The original waveform (3.12.a) is aligned with a diagram showing the tube interfaces where the free reflection coefficients occur (3.12.b). The evolution of the values of the free reflection coefficients is also shown (3.12.c).

From the diagram 3.12.b, it is clear that the topologies relate to the various waveform regimes characterizing the different phonemes of the utterance. Some invariance of the topology may be observed where the waveform remains stable. This diagram is also interesting in the sense that it recalls the notion of independent overlapping segments evoked in many speech production theories. More work would be needed to assess if and how the topologies found by the NUT scheme relate to the layout of the actual human speech production apparatus.

To go further, it would be interesting to study more precisely the variability of the topologies in relation to some specific phonetic classes. If the topologies prove stable for various classes of speech signals, the optimization tree could be pruned to retain only some speech-specific configurations. In a coding framework, this would allow to reduce the number of bits necessary to index the codebook of candidate topologies. To study that possibility, the NUT analysis should be performed with various instances of individual vowels and consonants, and the resulting configurations should be plotted into phoneme-specific diagrams.

We have not been able to put this experiment in practice because we have not disposed of a sufficient time and of sufficient resources to collect a suitable phonetic database. As a matter of fact, the existing databases usually contain whole words with a separate labeling information rather than sound files for each individual phoneme. Hence, an automatic phoneme



(a) Original waveform;

(b) places of free interfaces (unconstrained reflection coefficients);

(c) values of the reflection coefficients.

Figure 3.12: Results of a [8/24] NUT analysis on the French utterance *"dans cette crémerie"*.

extraction system should be used[10]. This extraction should rely on a hand-labeling of the sound files, since an automatic labeling, e.g., through forced alignment, determines recognition specific speech units rather than the phonemes defined by the human perception of minimal pairs. For example, it is common to hear pieces of the surrounding consonants in the "vowels" delimited by a forced alignment. This problem does not occur with hand made labeling. Given the overhead induced by those practical matters, the realization of this experiment has been left in the domain of future work.

Is there a global speech specific topology ?

An alternate experiment has consisted in learning a global NUT filter configuration from speech segments that span more than one frame, to find the layout which produces the least residual error averaged over all the frames. This layout may be informative about some speech production phenomenon that would not be visible in the classical reflection coefficients. This is also an alternate way to determine some NUT lattices which are globally specialized in the modeling of a class of signals instead of being specialized to one particular frame. Using these alternate class-specific topologies could again help pruning the search tree: instead of searching the total number $\binom{N_u-1}{N_s-1}$ of possible topologies, a reduced number of suboptimal but class-specific topologies could be searched.

To determine some global topologies, three different speech signals have been used. One is a portion of a French test sentence ("*dans cette crémerie*") spoken by a male speaker, recorded in an anechoic chamber and sampled at 32kHz. The two others are 3 unconnected words ("*dormitory school has*"), taken from the English word lists of the UW-XRMB database for a male speaker (jw11) and a female speaker (jw16), recorded in laboratory conditions and sampled at 21.739kHz.

The tables 3.3 and 3.4 give the optimal configuration for various constraints and for the three tested speech signals. The number of configurations to be tested for each constraint is also indicated. The interpretation of these results is difficult, because the only clear-cut tendency of the optimization is to locate a maximum of degrees of freedom towards the "lips" end of the tube, or equivalently towards the output of the synthesis filter. It would be tempting to remark that in the human vocal tract apparatus, the maximum mobility actually occurs in the mouth region. However, given our current level of knowledge of the model, it is difficult to tell to which extent this phenomenon relates to actual physiological properties, or to which extent it is an artifact of our selective correlation analysis.

As a matter of fact, our model allows to de-couple the number of degrees of freedom of the

[10]The HCopy application in HTK [Young et al., 1999] allows to extract a label-specific part from a sound file. Perl scripts reading the phonetic transcription and calling HCopy for each label could therefore be used to cut a whole file into all its label-specific pieces. Alternately, integrating a segmentation module to our NUT analysis software should not be too difficult.

Number of DoFs	Nb. of configurations	Male speaker/32 kHz
1	31	2/32:2,30.
2	465	3/32:1,1,30.
3	4'495	4/32:1,1,2,28.
4	31'465	5/32:1,1,2,2,26.
5	169'911	6/32:1,1,1,1,2,26.
...	> 700'000	
26	169'911	27/32:18x1,3,6x1,2,3.
27	31'465	28/32:19x1,2,6x1,2,3.
28	4'495	29/32:27x1,2,3.
29	465	30/32:29x1,3.
30	31	31/32:29x1,2,1.
31	1	32/32:32x1.
Constraints including minimum length:		
6 DoF min. length = 2	134'596	7/32:5x2,18,4.
7 DoF min. length = 2	245'157	8/32:6x2,16,4.

Table 3.3: Best configurations for various [n/32] constraints applied to the 32kHz French male speaker data.

Nb. of DoFs	Nb. of conf.	Speaker jw11 (m)	Speaker jw16 (f)
1	21	2/22:1,21.	2/22:1,21.
2	210	3/22:1,1,20.	3/22:1,1,20.
3	1'330	4/22:1,1,5,15.	4/22:1,1,2,18.
4	5'985	5/22:1,1,4,4,12.	5/22:1,1,2,2,16.
5	20'349	6/22:1,1,2,2,1,15.	6/22:1,1,1,1,2,16.
6	54'264	7/22:1,1,2,2,1,4,11.	7/22:1,1,1,1,2,2,14.
7	116'280	8/22:1,1,2,2,1,3,1,11.	8/22:1,1,1,1,2,2,2,12.
8	203'490	9/22:1,1,1,1,2,1,3,1,11.	9/22:1,1,1,1,1,1,2,2,12.
9	293'930	10/22:1,1,1,1,1,1,3,1,11.	10/22:1,1,1,1,1,1,2,1,1,12.
10	352'716	11/22:1,1,1,1,1,1,1,3,1,7,4.	11/22:1,1,1,1,1,1,1,1,1,1,12.
11	352'716	12/22:11x1,11.	12/22:10x1,7,5.
12	293'930	13/22:11x1,7,4.	13/22:10x1,2,5,5.
13	203'490	14/22:11x1,3,4,4.	14/22:10x1,2,2,3,5.
14	116'280	15/22:11x1,3,2,2,4.	15/22:10x1,2,2,2,1,5.
15	54'264	16/22:11x1,3,2,1,1,4.	16/22:10x1,2,2,2,1,3,2.
16	20'349	17/22:11x1,2,1,2,1,1,4.	17/22:10x1,2,2,1,1,1,3,2.
17	5'985	18/22:14x1,2,1,1,4.	18/22:12x1,2,1,1,1,3,2.
18	1'330	19/22:14x1,2,1,1,3,1.	19/22:17x1,3,2.
19	210	20/22:14x1,2,1,1,2,1,1.	20/22:17x1,3,1,1.
20	21	21/22:18x1,2,1,1.	21/22:18x1,2,1,1.
21	1	22/22:22x1.	22/22:22x1.

Table 3.4: Best configurations for various [n/22] constraints applied to data from two speakers of the UW-XRMB database.

production model from the dimensions of its acoustic counterpart. This constitutes a first step towards connecting the parametric speech modeling techniques inherited from Digital Signal Processing with some speech production models issued from phonetic theories. Our approach entails a new form of model dimensionality reduction, which proves efficient on the ground of spectral modeling accuracy. This reduction is based on the selection of the most important partial correlations, which are related to the interleaving of the frequencies making the speech signal. These frequencies are themselves shaped by the resonances of the vocal tract. However, a different kind of work, more related to the domain of the phonetic sciences, is still needed to determine the extent to which our selective partial correlation analysis is accurate as an interface between the acoustics and the actual physical configurations of the vocal tract.

3.4 Summary of the NUT inverse filtering analysis method

The NUT inverse filtering analysis method can be decomposed into the following steps :

1. **Signal preprocessing**

   (a) To remain consistent with speech production paradigms, **adapt the sample frequency of the speech data** to the constraint imposed by the model, namely :

   $$F_s = \frac{c}{2\,\Delta l_{unit}} \tag{3.24}$$

   where c is the speed of sound ($34000\,cm/s$) and Δl_{unit} is the unit length defined in the section 2.2.2.

   Assuming that a vocal tract is 17 centimeters long on average, the adequate sampling frequency corresponds to the number of unit sections considered in the NUT model, e.g. 32kHz for [n/32], or 24kHz for [n/24]. The resampling can be performed with the polyphase algorithm [Bellanger et al., 1976; Elliott, 1987], which combines computational efficiency with a good interpolation quality (little or no aliasing is introduced by this method).

   This step is optional if one does not care about the consistency with tube models, but is just interested in viewing the model as a constrained correlation analysis.

   (b) **pre-emphasize the obtained speech wave** by a simple differentiation.

   This step is performed to compensate for the effects introduced by the shape of the glottal waveform and by the radiation effect at the lips. These effects are not otherwise modeled by the NUT filters.

   This point is therefore linked with the notion of consistency with tube models. Nevertheless, from a pure Digital Signal Processing point of view, it can be shown that the pre-emphasis helps reaching better spectral estimates by reducing the spectral tilt and hence augmenting the global spectral contrast [Makhoul, 1975a].

2. **Estimation of the parameters and of the topology**

   Several variants can be implemented, depending upon the way of specifying the topology :

   (a) **imposition of an a-priori topology** –(fig. 3.3)– a particular a-priori topology can be considered better suited to the analysis of a whole range of data. The corresponding NUT filter can be used as a parametric model without any further topology optimization. In this case, as seen from figure 3.10, data which do not correspond to the range of the a-priori topology will be modeled with a low spectral accuracy.

   The inverse filtering parameters can be estimated every 10ms by computing the reflection coefficients k_i from one of the estimators of table 3.1. For speech, 25ms observation windows are usually employed.

   (b) **frame-by-frame topology optimization** –(fig. 3.4)– Alternately, one can seek the best topology for each frame of the analyzed signal. In this case, for a given frame, all the topologies respecting a given $[N_s/N_u]$ specification are used in turn, and their parameters are estimated as in the variant (a). Their respective accuracies are computed in terms of mean squared residual error (e.g., $MSE = \frac{1}{2N}\sum_{i=1}^{N}\left(\left(\epsilon_i^+(M)\right)^2 + \left(\epsilon_i^-(M)\right)^2\right)$), and the (topology,parameters) pair giving the least MSE is retained to represent the analyzed frame.

   (c) **global optimization of the topology over a class of signals** –(fig. 3.5)– it is also possible to determine the best topology pertaining to a particular class of signals, e.g., sentences from a particular speaker, or a particular class of speech sounds, and to apply it to unseen frames. This case requires two passes. The first pass allows to find the topology, out of a set respecting a given $[N_s/N_u]$ specification, which produces the least mean squared residual error cumulated over the whole set of "template" signal frames. The second pass uses the determined topology to extract the reflection coefficients on a frame-by-frame basis, without re-optimizing the topology for each frame.

   In all these variants, the extracted reflection coefficients can be further transformed into log area ratios or into area functions.

3. **Inverse-filtering of speech**

   The inverse filtering of speech, using the extracted models, allows to retrieve the residual excitation signal for a potential application to coding or synthesis.

This analysis scheme reduces the number of required LPC modeling parameters while keeping the related increase of prediction error to its lowest level. Equivalently, it allows to predict a signal sample from a longer portion of its past with fewer parameters than the usual all-pole models.

The corresponding reduction in the number of degrees of freedom represents a speech production constraint in the sense that if one assimilates speech production to acoustic filtering by a lossless tube, a relevant number of degrees of freedom can be imposed to the production model independently of the dimension of the acoustic counterpart. Besides, the $[N_s/N_u]$ specification accounts for a smoothness constraint on the tube shapes in the longitudinal direction : imposing some fixed zero-values, or equivalently "gluing" some interfaces, constrains the potentially irregular shapes of tubes with a great number of equal-length sections to adopt the smoother shapes of tubes with a reduced number of unequal-length sections. Finally, the optimization of the lengths of the sections allows to code part of the acoustic information as structural modifications of an underlying physical model.

It could be argued that from a geometric and physiologic point of view, NUT tubes rely on rather crude tube shapes. For instance, a more anthropomorphic constraint could be given in the form of reflection coefficients fixed to non-null values. These non-null constants could be drawn from the study of X-ray pictures or other adequate vocal tract shape measurements. The next chapter will expose our attempt at incorporating some more anthropomorphic articulatory constraints in the NUT/LPC framework.

Chapter 4

Speech analysis by projection into a basis of human shape factors

The model of Maeda [Maeda, 1979, 1990] represents the sagittal profile of the vocal tract rather than the whole tube shape. This profile is described by a linear combination of basis vectors that have been determined from the observation of an X-ray database [Maeda, 1979]. To recover the whole area function from the profiles, the $\alpha\beta$ transform [Heinz and Stevens, 1965; Sundberg, 1969; Perrier et al., 1992; Lecuit and Socquet, 1996] can be subsequently used. In the design of this model, morphologic issues come before acoustic or phonetic issues : the model first tries to be anthropomorphic, since it is based on the direct modeling of the movements of the human vocal tract profiles, and then it is verified that it provides an accurate acoustic modeling (i.e. correct formant trajectories, a realizable vowel space [Boë et al., 1989], etc.).

So far, attempts at inverting Maeda's model, i.e. recovering its parameters from some acoustic features, have been mainly based on the use of codebooks [Laprie and Mathieu, 1998; Ouni and Laprie, 2000] or of neural networks [Laboissière, 1992]. These approaches do not try to explain analytically the link which exists between the acoustic level and the articulatory level : they build and explore a mapping based on the simple training of an associative system. In the case of neural networks, an additional problem is that the mapping cannot be inverted in a straightforward way.

As an alternative, we propose a new idea which builds upon the analogy exposed in chapter 2. Our idea consists in interfacing the linear profile shape model with the inverse lattice filtering methods through the application of a least squares shape smoothing in the articulatory domain. Starting from the area functions issued by the LPC or the NUT analysis, this can be realized with the help of two additional analytic steps :

- the projection of the area function into the space of sagittal shapes through the analytic inversion of the $\alpha\beta$ transform;

- the smoothing of the obtained shapes through least-squares decomposition into a basis of shape factors drawn from Maeda's statistical analysis of human X-ray data.

This idea creates a link between the acoustic level and the articulatory level through a series of analytic and invertible relations.

In the following, this method will be designated by the acronym ReALiSM, standing for "Relating Acoustics to a Linear Shape Model". After having reviewed the various components of the corresponding processing chain, we will report some inversion results obtained with both synthetic and human vowels [Krstulović, 2000b].

4.1 Description of the components of the ReALiSM system

The system decomposes into the blocks depicted on figure 4.1. Each block will be described below.

4.1.1 Relation between the sound and the acoustic parameters

As seen in the previous chapters, inverse lattice filtering provides an interface between the sound of speech and some acoustic features. For the current preliminary system, a 7^{th} order unconstrained lattice filter has been used with speech sampled at 8kHz. Its 7 reflection coefficients have been estimated with the Itakura-Saito estimator. Alternately, a NUT model could be used at this stage.

In the synthesis direction, the generation of the speech signal corresponds to exciting a lattice filter with the residual error obtained after inversion, or with quantized error sequences (such as in Code Excited Linear Prediction, CELP) or with a white noise, a pulse train or a more elaborate synthetic glottal-like excitation (e.g. Rosenberg's glottal wave [Rosenberg, 1971]). The filter's parameters are updated every 10 milliseconds.

More elaborate and complex methods, based on aeroacoustics [Krane et al., 2000] or the Finite Element Method [Matsuzaki and Motoki, 2000; Motoki et al., 2000], are currently being developed to compute accurately the acoustic filtering effects of some non-cylindrical vocal tract shapes. Alternately, a more simple and well understood method [Flanagan, 1972; Mrayati, 1976; Atal et al., 1978; Badin and Fant, 1984, etc.] is commonly used to find the acoustic counterpart of the area function with an acceptable accuracy. This method considers that the acoustic effects are not significantly disturbed if the vocal tract is assimilated to a series of lossy cylindrical or conical section. The area function describes circular areas, and an electrical analogy is used to formalize the computation of the transfer function. The next simplification step, which considers

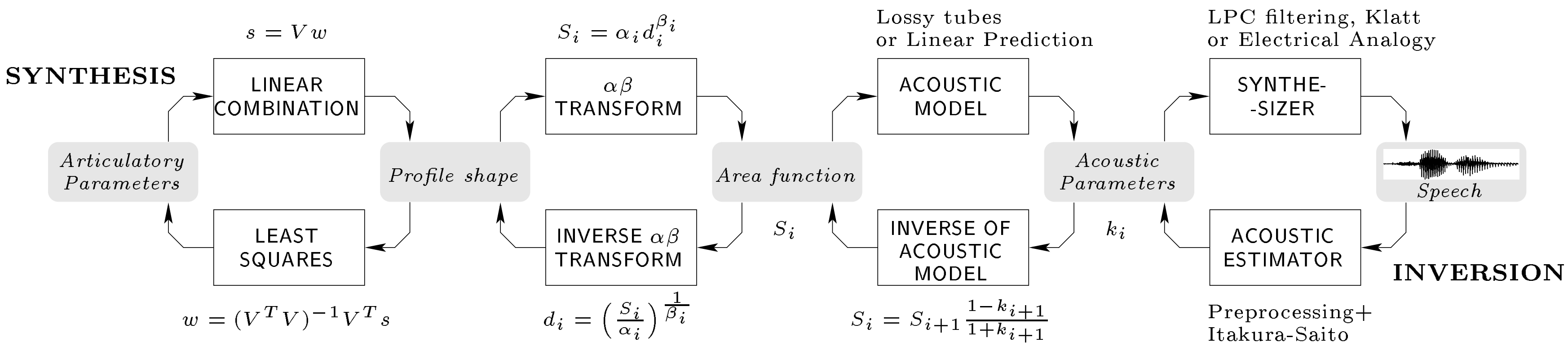


Figure 4.1: Block diagram of the ReALiSM system.

that the energy losses due to viscosity, heat conduction and wall yielding can be neglected, leads to the lattice filtering formulation.

The more elaborate acoustic models cited above can be plugged in place of the employed lossless All-Pole model, provided they admit an inversion method to preserve the integration of inversion and synthesis in a coherent processing framework.

4.1.2 Relation between the acoustic parameters and the area function

As discussed in chapter 2, the process of All-Pole filtering is analogous to acoustic filtering in discrete lossless acoustic tubes provided that :

1. sound waves are considered to be plane fluid waves;
2. the lengths of the individual tube sections are kept short compared to the wavelength at the highest frequency of interest (this introduces a spectral boundary);
3. the sampling rate of the speech signal is $F_s = \frac{c}{2\Delta l_{unit}}$, where Δl_{unit} is the length of a tube section;
4. no losses are accounted for.

As exposed in the previous chapters, if the speech signal is pre-emphasized to compensate for the spectral characteristics of the glottal excitation and for the radiation impedance at the lips, estimates of the lossless tubes' area functions can be recovered from the speech waveform by using inverse lattice filtering and the following relation :

$$k_{i+1} = \frac{S_{i+1} - S_i}{S_{i+1} + S_i} \quad \Leftrightarrow \quad S_i = S_{i+1} \frac{1 - k_{i+1}}{1 + k_{i+1}} \tag{4.1}$$

where k_{i+1} denotes the reflection coefficients and S_i denotes the areas of the corresponding discrete lossless tube, numbered in ascending order from lips to glottis. If the lips section is not available, this recursion can be applied in the reverse direction by considering that the glottis section is fixed to $1.5cm^2$.

Considering that the area function (or vocal tract) should be approximately 17 centimeters long, and that the speed of sound is equal to 340m/s, the 3$^{\mathrm{rd}}$ condition imposes to use M sections for speech sampled at M kiloHertz. In the preliminary system, 8 sections, or equivalently 7 mobile interfaces, have been used for speech sampled at 8kHz (this imposes the 7^{th} order filter employed in the LPC analysis of section 4.1.1).

The area function can be resampled to meet further processing requirements. In our case, the 8 sections corresponding to the 7^{th} order LPC model have been redistributed over 30 sections to match the dimensions of the $\alpha\beta$ transform and of the profile shape model described in the upcoming sections. Alternately, a 30$^{\mathrm{th}}$ order LPC or a [n/30] NUT model could be used with speech sampled at 30kHz.

4.1.3 Connecting the areas and the profiles

Since the human vocal tract does not have circular sections, the relation between area functions and vocal tract profiles is usually described by the $\alpha\beta$ transform [Sundberg, 1969] :

$$S_i = \alpha_i d_i^{\beta_i} \quad \Leftrightarrow \quad d_i = \left(\frac{S_i}{\alpha_i}\right)^{1/\beta_i} \tag{4.2}$$

where S_i is the area of a section, d_i is the diameter measured from the profile outline, and (α_i, β_i) are section-dependent parameters drawn from the study of human vocal tract shapes [Sundberg, 1969; Perrier et al., 1992]. From (4.2), it is clear that this relation admits an exact, one-to-one reciprocal.

Various definitions exist for the diameters d_i. While the works related to Maeda's model usually employ a pseudo-diameter derived from some lateral areas, our choice has been to stick to the original $\alpha\beta$ rationale by measuring the diameters along the lines of a semipolar grid. However, finding adequate transformations between the areas and the profiles is still an active field of research: numerous other transformations exist [Lecuit and Socquet, 1996]. Other models can readily replace the original $\alpha\beta$ relation in the processing chain of figure 4.1 if they prove to be more accurate and still invertible. Alternately, this step could be suppressed by directly using a three dimensional factorial model, as the one proposed by Badin et al. [1998].

4.1.4 Description of the profiles with a linear articulatory model

Principle of the original model

Maeda's model [Maeda, 1979] represents the vocal tract profile shapes as a linear combination of articulatory factors. These factors are determined from some measurements taken on X-ray movies showing the vocal tract of a real speaker, as shown on figure 4.2.

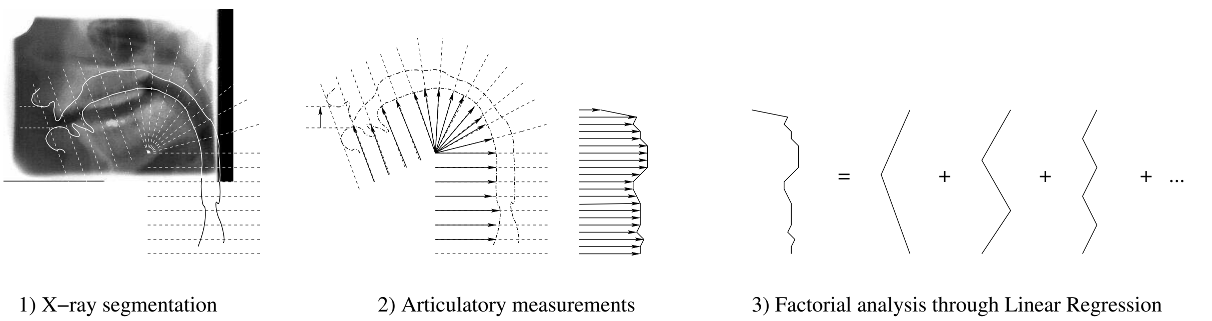


Figure 4.2: Design of Maeda's model.

First, the X-ray images have been hand-segmented to extract the vocal tract contours. From these contours, 32 measurements made in 3 distinct zones of the vocal tract are taken, with the help of a semipolar grid (fig. 4.3) :

- in the lips zone, lip aperture ($\mathrm{LIP}_{\mathrm{ap}}$), lip protrusion ($\mathrm{LIP}_{\mathrm{pr}}$) and lip width ($\mathrm{LIP}_{\mathrm{wd}}$) are measured;
- in the tongue region, 25 tongue shape measures are plotted along the semipolar grid lines (TNG_1 to TNG_{25});
- in the larynx zone, two points delimiting the lower larynx edge are plotted (LRX_{x_1,y_1} and LRX_{x_2,y_2}).

In addition, a fixed back wall outline is measured in the semipolar grid. It delimits the diameters in the lips and the tongue regions.

Each of the 32 mobile features is related to a set of 5 control parameters by an orthogonal factor analysis (a form of driven linear regression) [Maeda, 1979]. The linear factors are determined so that the control parameters have an articulatory interpretation :

- jw represents the influence of the jaw on all the features;
- tp, ts and tt control the tongue position, the tongue shape and the tongue tip position respectively;
- lh and lp control the lip height and the lip protrusion;
- lx controls the larynx height.

In this framework, modeling a vocal tract inner contour corresponds to applying the following block-structured linear equation system of dimensions 32×7 :

$$\left[\begin{array}{c|ccc|cc|c}
l_{1,1} & & & & l_{1,2} & l_{1,3} & \\
l_{2,1} & & 0 & & l_{2,2} & l_{2,3} & 0 \\
l_{3,1} & & & & l_{3,2} & l_{3,3} & \\
\hline
t_{1,1} & t_{1,2} & t_{1,3} & t_{1,4} & & & \\
t_{2,1} & t_{2,2} & t_{2,3} & t_{2,4} & & 0 & \\
\vdots & \vdots & \vdots & \vdots & & & \\
t_{25,1} & t_{25,2} & t_{25,3} & t_{25,4} & & & \\
\hline
la_{1,1} & & & & & & la_{1,2} \\
la_{2,1} & & 0 & & & & la_{2,2} \\
la_{3,1} & & & & & & la_{3,2} \\
la_{4,1} & & & & & & la_{4,2}
\end{array}\right]
\left[\begin{array}{c} jw \\ \hline tp \\ ts \\ tt \\ \hline lh \\ lp \\ \hline lx \end{array}\right]
=
\left[\begin{array}{c} \mathrm{LIP}_{\mathrm{pr}} \\ \mathrm{LIP}_{\mathrm{ap}} \\ \mathrm{LIP}_{\mathrm{wd}} \\ \hline \mathrm{TNG}_1 \\ \mathrm{TNG}_2 \\ \vdots \\ \mathrm{TNG}_{25} \\ \hline \mathrm{LRX}_{x_1} \\ \mathrm{LRX}_{y_1} \\ \mathrm{LRX}_{x_2} \\ \mathrm{LRX}_{y_2} \end{array}\right] \tag{4.3}$$

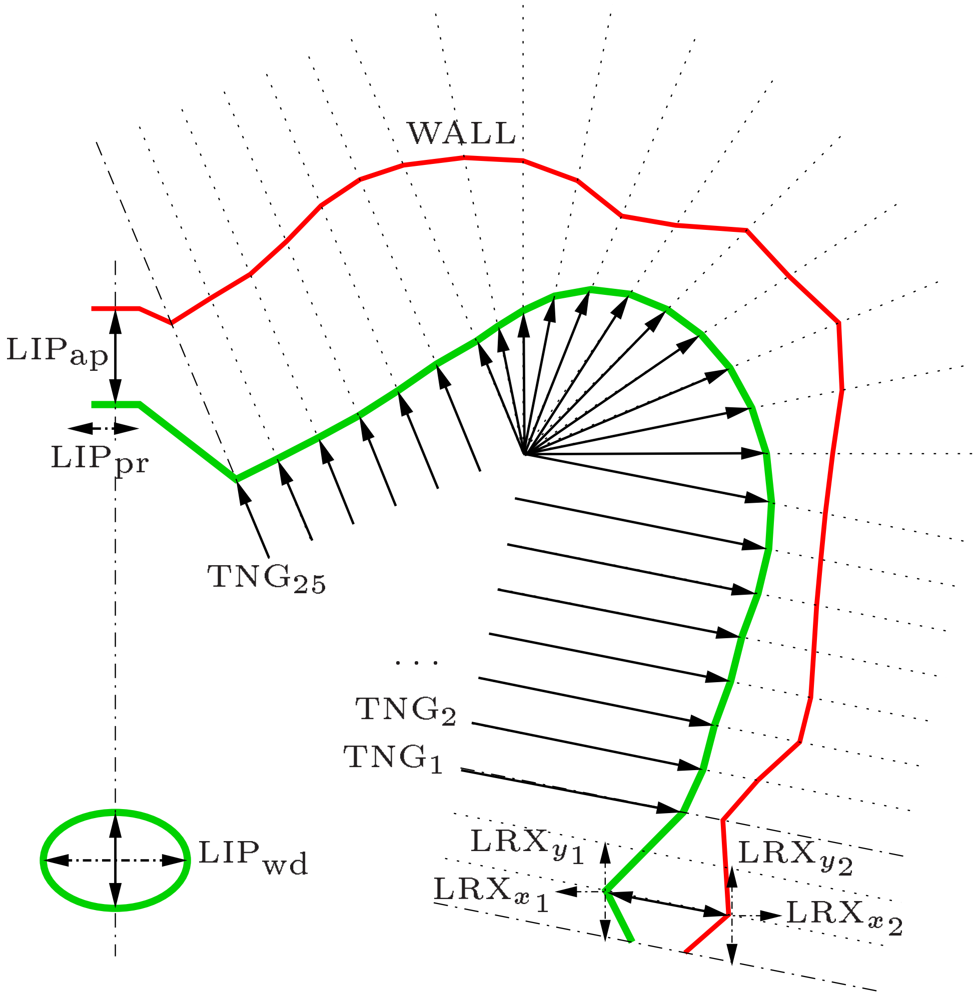


Figure 4.3: The various components of the linear profile shape model.

where $l_{i,j}$, $t_{i,j}$ and $la_{i,j}$ are the factors determined for the lips, the tongue and the larynx action respectively. The features resulting from this linear combination are projected back to the semipolar grid to obtain the actual vocal tract contour.

Fixed length approximation to the model

The area function, and thus the profile, must have a fixed length to permit a connection with lattice filtering (or, more exactly, to be compatible with a fixed sampling frequency). Hence, our system makes use of some approximations:

- the lip protrusion parameter lp and the related factors are ignored. The lip protrusion measure $\mathrm{LIP}_{\mathrm{pr}}$ is fixed to 0.5cm instead of being computed from the linear combination;
- the lip width quantity $\mathrm{LIP}_{\mathrm{wd}}$ is also ignored: the area of the lips section is derived from the lip aperture parameter by direct application of the $\alpha\beta$ transform;
- the larynx shape is approximated by assuming that the two larynx points LRX_{x_1,y_1} and LRX_{x_2,y_2} always move along the second grid line. The larynx height parameter lx is not used;
- a fixed glottis section is added along the last grid line, with a constant area of $1.5cm^2$.

The subsequent lines and columns are thus removed from (4.3), reducing the dimensions of the system to 30×5:

$$\left[\begin{array}{c|ccc|c} l_{2,1} & & 0 & & l_{2,2} \\ \hline t_{1,1} & t_{1,2} & t_{1,3} & t_{1,4} & \\ t_{2,1} & t_{2,2} & t_{2,3} & t_{2,4} & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ t_{25,1} & t_{25,2} & t_{25,3} & t_{25,4} & \\ \hline la_{1,1} & & & & \\ la_{2,1} & & 0 & & \\ la_{3,1} & & & & \\ la_{4,1} & & & & \end{array}\right] \left[\begin{array}{c} jw \\ \hline tp \\ ts \\ tt \\ \hline lh \end{array}\right] = \left[\begin{array}{c} \mathrm{LIP}_{\mathrm{ap}} \\ \hline \mathrm{TNG}_1 \\ \mathrm{TNG}_2 \\ \vdots \\ \mathrm{TNG}_{25} \\ \hline \mathrm{LRX}_{x_1} \\ \mathrm{LRX}_{y_1} \\ \mathrm{LRX}_{x_2} \\ \mathrm{LRX}_{y_2} \end{array}\right] \qquad (4.4)$$

This approximation does not affect the tongue shapes, but affects the overall vocal tract length through blocking the lip protrusion and the larynx height variations. This is likely to entail some inaccuracies in the synthesis and inversion of some French vowels, such as /u/ and /y/[1], for which the lip protrusion plays an important role.

[1] The Worldbet phonetic notation [Hieronymus, 1993; Lander, 1997] is used throughout the present report to identify the vowels.

Inversion

The linear system (4.4) comprises 30 equations for 5 variables. Hence, inverting it, i.e. finding the values of the control parameters given a particular shape, amounts to solving an overdetermined linear problem. Solutions for such problems are available through Least-Squares solving:

$$Vw = s \quad \rightsquigarrow \quad \hat{w} = (V^T V)^{-1} V^T s \tag{4.5}$$

where V is the matrix of known factors, s is the vector describing the tract shape, and w is the vector of articulatory parameters ($\hat{w}$ being its Least Squares estimate). This method finds the closest shape, in a Least Squares sense, that the linear model can produce with respect to the given shape. Hence, it performs a model-driven smoothing of the input shape.

Fortunately, in the fixed length case, the factors matrix V has full rank and is well conditioned. Numerous algorithms such as the Singular Value Decomposition (SVD) and the QR factorization are thus available to solve the problem [Golub and Van Loan, 1983]. We have used the SVD in the current implementation (LAPACK routine `dgelss` [Anderson et al., 1999]), but it could be substituted with any other algorithm liable to bring more adapted or more accurate solutions.

4.2 Acoustico-articulatory inversion results

4.2.1 Auto-inversion

As a first assessment, it is useful to verify whether information losses that occur within Least-Squares smoothing, area functions resampling and reflection coefficients estimation still allow for the recovery of synthetic template tract shapes.

Hence, a set of synthetic vowels has been produced, using articulatory parameters that correspond to some cardinal French vowels registered in the UPSID phonemic database [Maddieson, 1984]. Informal listening tests ensured that the synthetic vowels were acceptable in spite of the fixed length approximation and the lossless LPC synthesizer. A more complete evaluation of the synthesis capabilities of the system should nevertheless be performed, e.g. with formant measurements and comparison with human values.

Results depicted in figure 4.4 show that the estimated shapes lie close to the original synthetic shapes. The observed variations result from the pulse train excitation used for synthesis.

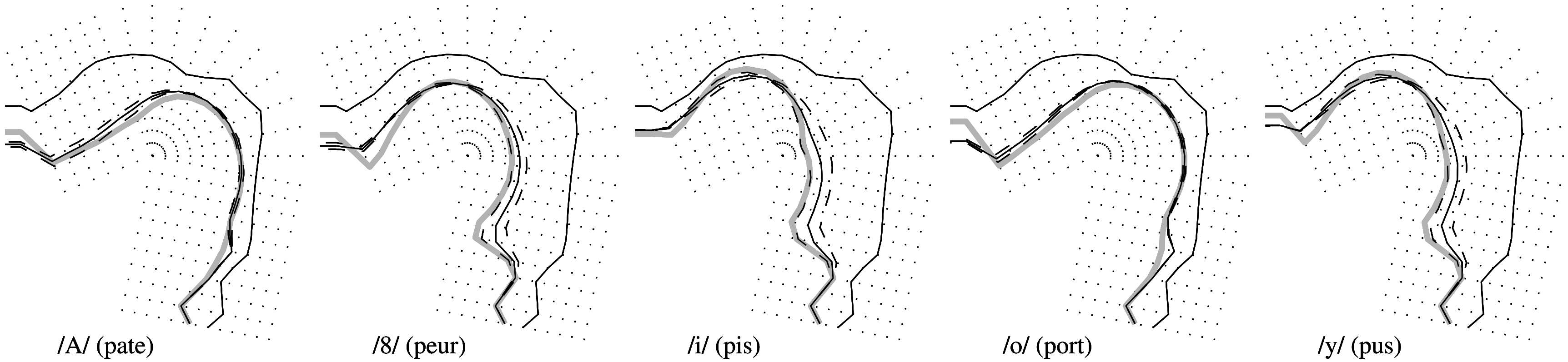


Figure 4.4: Auto-inversion for 5 synthetic vowels. The bold grey line corresponds to the original profile. The thin black line shows the mean of the extracted profiles, which vary between the two dashed lines.

4.2.2 Inversion of real speech

The system has been used to invert some real speech recorded from a French male speaker in a quiet environment. Several vowel sequences and VCV sequences have been tested. Results corresponding to a "vocalic triangle" (/i e E A o u/ sequence) and to the /A b i/ sequence are given in figures 4.5 and 4.6 (next page). The system appears to locate the cavities at phonetically relevant places of articulation (e.g. front for /A/, back for /i/ [Ladefoged and Maddieson, 1996]). Lip apertures also seem realistic. In the /A b i/ sequence, the /b/ consonantal closure appears to be detected. A spurious constriction is nevertheless observed in some cases at the back of the tongue.

The obtained results are good from a qualitative point of view. Further work would include comparing them with human data to assess their quantitative accuracy.

4.3 Discussion

4.3.1 Principle of the ReALiSM estimation

The ReALiSM scheme minimizes an articulatory cost on shapes derived after minimization of an acoustic criterion. Equivalently, it allows to drive the parameters of the linear model with the outcome of the LPC analysis. It could be argued that this somewhat violates the assumption that speech is an acoustic object carrying inherent articulatory constraints, since the acoustic estimation is corrected a posteriori instead of being directly constrained by the shape model. As a matter of fact, the coupled evolutions of the acoustics and of the articulatory model can be interpreted as a latent space modeling problem. When the mathematical relation linking the observations and the latent variables complicates too much the optimization in the latent space[2], such sub-optimal solutions are commonly used.

On the other hand, being able to add some articulatory information coming from an external model allows to go beyond the limitations of speech analysis in terms of acoustics only. As remarked by Sondhi during the discussions related in [Carré et al., 1977, p.21] :

> "(...) if we go back from the speech wave to the articulatory parameters, then we are in a vicious circle which we can never break, because the articulatory information cannot tell you anymore than what the speech waves carries about the articulation. We have to break the circle somewhere and the only way to do it is to make a measurement, otherwise the information that we get is a different parametric representation of exactly the same information."

[2] We have tried to incorporate the expression of the linear model in an expansion of Burg's error criterion (the equation (3.1) of section 3.1.1), to solve for a direct estimator of Maeda's parameters. This leads indeed to a very complicated mathematical problem.

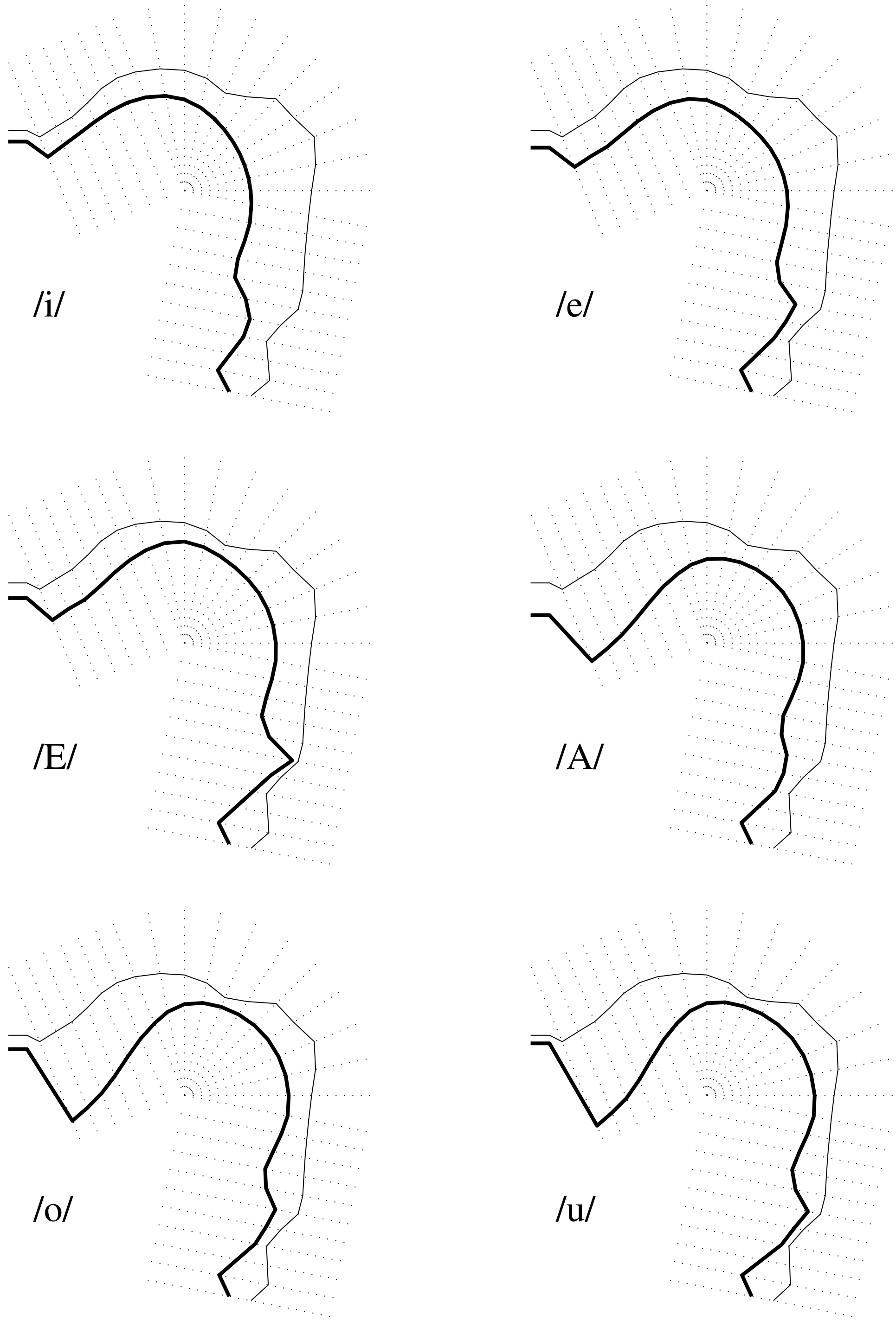


Figure 4.5: Inversion of human vowels.

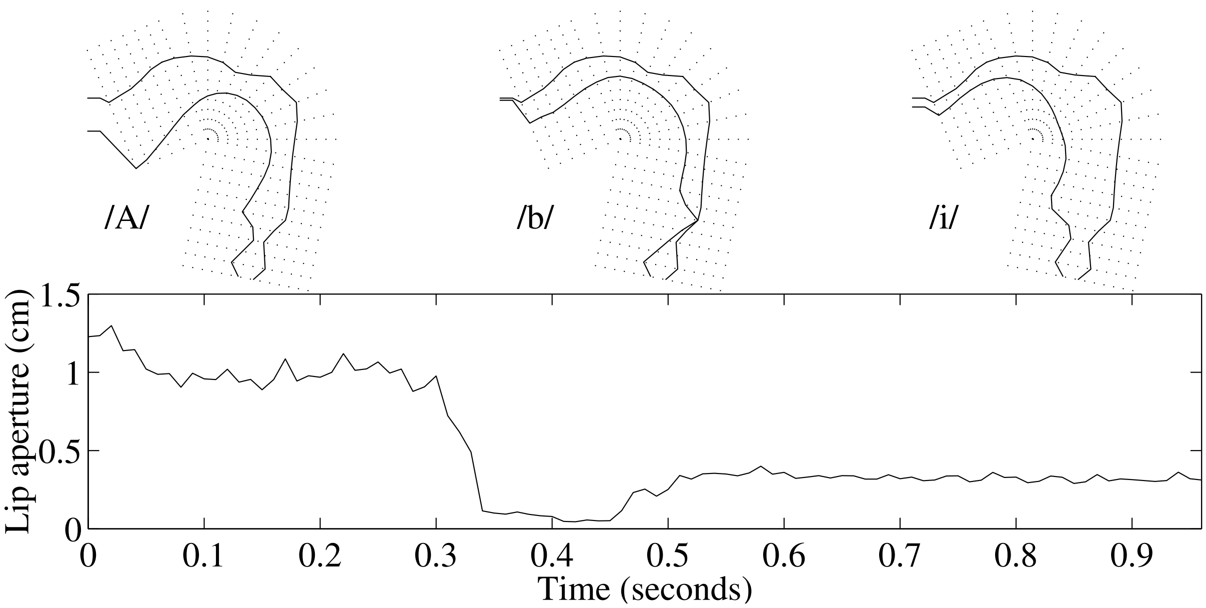


Figure 4.6: Inversion of /abi/.

The measurements proposed by Sondhi were referring to the capture of the vocal tract impedance at the lips, which can be used to recover the whole vocal tract shape [Sondhi, 1979]. In our case, the principle of the underlying model is very different, but the idea of adding some relevant articulatory information from a separate source is present.

4.3.2 Articulatory relevance of the LPC solution

Alternately, our results bring support to the idea that LPC is able to capture some pieces of qualitatively relevant articulatory information, provided an adequate post-processing scheme. The addition of the anti-$\alpha\beta$ transform and of the least-squares shape smoothing opens the door to a true comparison of the LPC-based inversion with some actual human vocal tract measures.

However, in the absence of such a comparison, no conclusion can be drawn yet as to the anthropomorphic accuracy of the extracted features. Such a comparison is still relatively difficult to perform in practice, with regard to the cost of acquiring and segmenting articulatory data and with regard to the fact that a method to normalize the measured vocal tract shapes without betraying the corresponding acoustics is not available.

Nevertheless, if the method eventually proves to provide a sufficiently precise acoustico-articulatory inversion (AAI) result, it will represent a significant advance in the domain of AAI since it is able to operate in real time. To help the method getting closer to actual AAI, any of the blocks used in the acoustico-articulatory chain of figure 4.1 can be replaced with a more elaborate or more precise component, provided that the replacement block admits a reciprocal. Hence, current limitations may be alleviated in future versions by the use of a more detailed profile shape model, better profile-to-area transformations, or acoustic estimators incorporating more elaborate relations to speech production.

4.3.3 Bridging the gap between Articulatory Modeling and Digital Signal Processing

Globally, the method creates a straightforward link between the shape-based articulatory model and the whole gear of parametric DSP tools (all-pole spectral modeling, spectral distortion measures, parametric speech coding methods, etc.). This link could be exploited in the framework of some Automatic Speech Processing (ASP) applications.

The next development step towards a use in ASP would consist in estimating some smoothed profile shapes from sound, going back to the spectral modeling level and then examining the spectral distortions induced by the least squares shape smoothing (figure 4.7). Some additional processing could be applied in the articulatory domain to allow for some articulation-based warping of the speech spectra.

We have begun to investigate these points, but have faced large spectral distortions. More work is needed to determine the exact cause of these distortions, and to tell if they constitute a meaningful or meaningless warping of the spectra. Only after a better understanding of this point will it be reasonable to compute speech features for ASP in the ReALiSM framework.

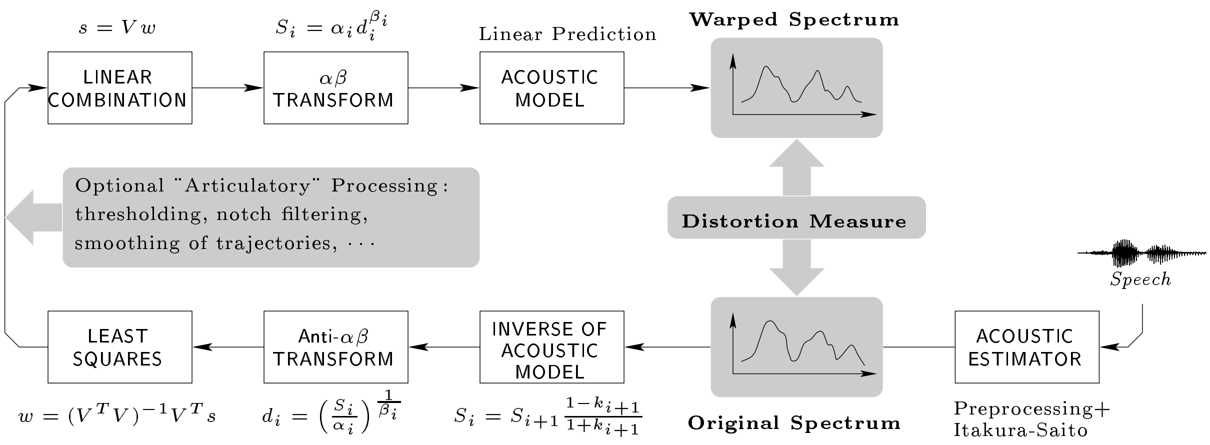


Figure 4.7: Evaluation of the spectral distortion in the ReALiSM framework.

Chapter 5

Potential applications

The NUT and ReALiSM analysis schemes match some speech production paradigms with the LPC speech modeling method, which is the core of most of the speech feature extraction systems used in state of the art Automatic Speech Processing (ASP) applications. Hence, the NUT and ReALiSM methods represent an innovative contribution to the realization of production based ASP. The present chapter exposes some research tracks to develop these methods in the areas of speech recognition, speech coding and speech enhancement.

5.1 Speech recognition

Speech recognition experiments from other researchers bring support to the idea of using articulation in speech processing: they are discussed in section 5.1.1. However, a critique of these experiments is necessary before implementing a functional articulation-based recognition system. In particular, the question of defining a perceptually relevant articulatory distance measure has to be evoked: this is done in section 5.1.2. NUT models propose a straightforward solution to this problem, and preliminary speech recognition results using NUT analysis are presented in section 5.1.3.

5.1.1 Speech recognition with articulatory paradigms: results from other researchers

Some authors have suggested that speech recognition would significantly benefit from the application of the Articulatory Phonology theory [Schroeter and Sondhi, 1989; Rose et al., 1994]. As an application of this idea, some speech and speaker recognition systems have been developed by many researchers. These systems can be divided in two classes:

- those which model the probability densities of articulatory features alone or mixed with pure acoustic features, in order to perform a direct recognition of the articulatory features;
- those which use some articulatory information to structure the acoustic probability density models.

Recognition of articulatory features

Schmidbauer [1989] has found that the performance of HMM-based recognizers is improved by introducing an articulatory feature vector as an intermediate level between the acoustic and the phonetic level in the decoding scheme. He observed a 9.8% relative decrease in error on a speaker dependent phoneme recognition task, as compared with a system using plain Mel Cepstrum Coefficients (MFCCs).

Zlokarnik [1993] has reported up to a 70.7% relative decrease of the Word Error Rate (WER) in a speaker dependent, highly-confusible nonsense words recognition experiment, by augmenting the usual MFCC feature vectors with articulatory data recorded by an electromagnetic articulograph (EMA) [Cartsens and Carstens, 2001]. He also tried recognition with some articulatory features inferred from the acoustics by means of a neural network, but could not obtain an improvement in WER with the inferred features. From this work, he therefore drew two important conclusions:

- the articulatory data brings an information that is complementary to the acoustic information;
- while the use of natural[1] articulatory features does bring an improvement of recognition scores, the quality of the acoustic-to-articulatory inversion scheme has to be carefully checked to reach such an improvement.

Papcun et al. [1992] correctly recognized a set of highly confusible consonants ({/p/,/b/}, {/t/,/d/}, {/k/,/g/}) using a template matching technique which was using some observations of the articulatory trajectories alone. These trajectories were estimated by a neural network trained on the University of Wisconsin (UW-XRMB) database [Westbury, 1994]. In a follow up of this work, Zacks and Thomas [1994] have obtain a 86.67% recognition rate on a speaker independent vowel recognition task, using neural networks trained with estimated gestural features alone. For the acoustico-articulatory inversion task, they proposed to use a new neural network architecture, able to recover trajectories from sound after training on the UW-XRMB database. They complemented the network with a template matching algorithm to go back from the trajectories to a gestural level, thus being in closer accordance with the theory of articulatory phonology.

[1] i.e. measured from humans, as opposed to features estimated from inversion.

Globally, this class of methods has only been applied to limited tasks such as phoneme or nonsense words recognition, and most of the time in a speaker dependent way. This is due to the difficulty in obtaining sufficient acoustico-articulatory data for a proper training of the models. As a consequence of the development of the new and more adequate MOCHA acoustico-articulatory database [Wrench and Hardcastle, 2000], Wrench and Richmond [2000] have been able to extend the principle of recognition with recorded articulatory features to a speaker dependent connected words recognition task. For this more realistic task, they observed a 6% relative improvement in Word Error Rate (WER).

Articulatory structure in acoustic models

Erler and Deng [1992] have developed a specific statistical approach, closely related to the Articulatory Phonology theory, in which the acoustic structure of the phoneme models reflects the organization of some underlying, possibly overlapping, articulatory features. They have successfully applied the idea to speaker dependent isolated nonsense word recognition, observing a 26% relative drop in WER for Consonant-Vowel (CV) nonsense words and a 13% relative WER reduction for CVC nonsense words as compared to classical phoneme models. Deng and Sun [1994] have further tested the idea in a speaker independent phoneme classification task and have reported up to a 27% relative WER reduction.

When using the same method on a large vocabulary, speaker dependent, isolated word recognition task, Erler and Freeman [1996] have observed that this Articulatory Feature Model (AFM) was competitive with conventional HMMs using MFCCs, despite the early stage of development of the AFM model.

In a follow up, Richardson et al. [2000a,b] have extended the recognition task to a speaker independent, isolated word, noisy telephone speech recognition task. While their Hidden-Articulatory Markov Models (HAMM) do not bring WER improvements when used alone, they help reducing the WER by 18-35% (depending on the lexicon size) when combined with some standard HMMs using MFCCs. In noisy speech, a relative improvement of 23-26% has been observed.

Frankel and King [2001] propose a radically different approach by modeling a latent articulatory state space as the hidden space for some Kalman filters (also known as Linear Dynamic Models, LDMs). These segmental models are subsequently gathered in Markov chains, thanks to a bigram language model, to allow for speech recognition. In a preliminary phone classification experiment, they compared the performances pertaining to different observation spaces. The use of recorded articulatory features (the EMA measurements of the MOCHA database [Wrench and Hardcastle, 2000]), in addition to cepstral coefficients plus energy, brought a 13% relative classification score improvement over the cepstral coefficients + energy alone. Conversely, the use of articulatory features simulated from a neural network did not bring an improvement.

(This confirms the observations of Zlokarnik [1993].) Frankel and King's research team is currently implementing a stack decoder adapted to their Linear Dynamic Markov models, to be able to apply the system to actual speech recognition problems (i.e., to be able to deliver a segmentation of the utterances rather than a plain phone classification).

Most of the works exposed in this paragraph require the use of *recorded* or *hand labeled* articulatory features rather than features estimated from sound. But gathering and labeling an articulatory database is an expensive and long running task. This explains to some extent why the principle of articulatory based speech recognition is still experimental and is just beginning to be tested in full-scale recognition systems. As evoked earlier, the development of full scale systems is tied to the development of a more adequate database.

But the difficulty of exploiting articulation in speech recognition is also due to the fact that one crucial question, that of the definition of an articulatory distance that would be meaningful with respect to speech perception, hasn't yet been solved.

5.1.2 Speech recognition with speech production paradigms : our approach

As a matter of fact, speech recognition techniques belong to the domain of *pattern matching*. In this respect, they require the definition of a distance between the features and the pattern or model to be matched. Whereas several acoustical distances have already been defined, put in relation to human perceptual properties and well studied [Gray and Markel, 1976; Rabiner and Juang, 1993], no *perceptually significant articulatory distance* has been defined.

For instance, it can be shown that the Euclidean distance between cepstral coefficients is related to the root mean square log-spectral difference [Basseville, 1989; Rabiner and Juang, 1993], which in turn is perceptually relevant [Gray and Markel, 1976]. Furthermore, the log-likelihood computed from HMMs can be viewed as a Mahalanobis distance between a template Gaussian probability density function and an observation vector [Basseville, 1989; Krstulović, 2001a]. The Mahalanobis distance is in turn equivalent to a Euclidean distance weighted by a covariance matrix to allow for a margin of variability in the observations. Hence, decoding cepstral observation sequences with respect to HMMs comprising multi-Gaussian densities in their states amounts to operating a series of perceptually relevant spectral distortion measures, within a certain margin of variability, and accounting for the transitional constraints imposed by a hidden Markovian state space.

This interpretation should be carefully re-evaluated when using HMMs to model other features than cepstral coefficients. As stated in [Rabiner and Juang, 1993] (p.191) :

> "For other parametric representations [than cepstra], such as log area ratio coefficients, a Euclidean distance $[\cdots]$ might actually be reasonable. However, spectral distortions based on parameters such as log area ratios have not been extensively

studied because it is relatively difficult to interpret the measured distortion in terms of spectral deviation."

This implies that articulatory distances may exist (for instance, a Euclidean distance between two vocal tract area functions or between two sets of log area ratios), but understanding their implications in the acoustic domain is rendered difficult by the limited knowledge of the topology of the articulatory-acoustic mapping. More clearly put, two vocal tract shapes that are close with respect to the Euclidean distance could result in very different acoustic results from an acoustic point of view.

As a matter of fact, speech recognition results obtained with some area functions or some log area ratios are usually worse than those obtained with cepstral features (see for instance [Atal, 1974] in the case of speaker recognition). The recognition results obtained with cepstral coefficients (versus other features) are usually the best in today's state-of-the-art systems, and Mel Frequency Cepstral Coefficients (MFCCs) have become a sort of *de facto* speech featuring standard.

Our work establishes an alternative approach, which consists in estimating some acoustic features constrained by speech production considerations instead of trying to extract and to recognize some purely articulatory features. Hence, we stay consistent with the consideration of speech as an "acoustic object". For instance, we remain consistent with telephone-based applications, where no visual or explicit articulatory clues are accessible. From the production constraints introduced in the inverse filtering framework, it is possible to estimate constrained reflection coefficients, then it is possible to derive the corresponding prediction coefficients, then finally to derive some Linear Prediction Cepstral Coefficients (LPCC) that inherit the information pertaining to the production constraints. The matching of the acoustic modeling with speech production modeling is operated in an integrated DSP framework, which allows to use meaningful acoustic distances.

In addition, since our systems do not require the use of recorded articulatory features, it is possible to perform full-scale speech recognition with usual, purely auditory speech recognition benchmarking databases.

5.1.3 Preliminary results

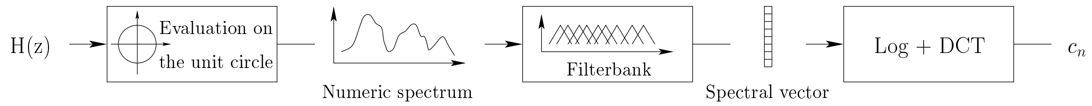
As stated in the previous section, it is now widely recognized that the best speech or speaker recognition results are obtained with cepstral features rather than with plain spectral features. The cepstral coefficients can be derived from the Linear Prediction coefficients in several ways:

- analytically, by application of the following recursion [Rabiner and Juang, 1993]:

$$c_n = -a_n - \frac{1}{n}\sum_{i=1}^{n-1}(n-i)a_i c_{n-i} \quad \text{for } n > 0 \tag{5.1}$$

where c_n are the cepstral coefficients and a_i are the linear prediction coefficients with $a_0 = 1$, $a_i = 0$ for $i > n$, and c_0 is the log of the spectral power;

- numerically, by computing the LPC smoothed spectra, integrating them in linear or Mel scale filterbanks and applying the Discrete Cosine Transform to the log of the outputs :



Since the LP coefficients a_i can be obtained from the reflection coefficients k_p through the relation (2.3) (page 12), the computation of some cepstral coefficients constrained by speech production considerations is straightforward in the framework of the NUT or ReALiSM analysis methods.

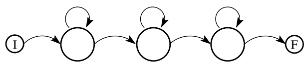
Experimental setup

In a preliminary speech recognition experiment, NUT-related cepstral features, computed numerically with the recursion (5.1), have been tested on a speaker independent, unconnected words recognition task. The training and test data have been drawn from the University of Wisconsin XRMB database [Westbury, 1994], comprising speech together with recorded articulation (see appendix A for the complete specifications of the database).

The choice of the UW-XRMB data had been initially motivated by the possibility to compare the extracted articulatory clues with some real ones, if we had managed to solve the distance problem, or to use some recorded articulation for recognition, although this was not our primary goal. The main advantage of this database is that it has become a reference for studies in speech production and its implications in speech recognition (see for instance [Papcun et al., 1992]). As a matter of fact, before the recent advent of the MOCHA database (which is still in a development phase [Wrench and Hardcastle, 2000]), the UW-XRMB database was representing the biggest volume of acoustico-articulatory data collected within a unique and well-defined measurement and segmentation framework.

Although the volume and repartition of this data is too limited to train speaker dependent models or to perform Large Vocabulary Speech Recognition (LVCSR), it is compatible with a medium vocabulary speaker independent speech recognition experiment. The specifications of our implementation of this experiment are given in table 5.1. An extensive prior work of resource management, including listening assessment and correction of the orthographic transcriptions, has been necessary and is described in the appendix A.

EXPERIMENTAL SPEECH RECOGNITION SETUP

- *Characteristics of the recognition task*: speaker independent, medium vocabulary, lists of isolated words, microphone speech with band-limited machine noise.
- *Base software platform*: HTK [Young et al., 1999].
- *Phone set*: DARPABET, 39 phonemes (same as the DARPA Resource Management database phone set), plus beep, silence, preamble and short pause (total 43 models).
- *HMMs type*: monophones, left-right, 3 emitting states. 

- *HMMs Emission PDF type*: multi-Gaussians, 5 Gaussian components, diagonal covariances.
- *Training set*: 41 speakers, approx. 12000 isolated words from the UW-XRMB "citation" tasks.
- *Test set*: 6 speakers, 1873 isolated words from the UW-XRMB "citation" tasks.
- *Lexicon*: part of CMU version 0.4; 106 words plus beep, preamble/postamble and silence.
- *Grammar*: beep, then preamble, then 5 to 7 words, then optional postamble.
- *Waveform characteristics*: 16 bits PCM, 21.739kHz sampling frequency, peak of machine noise around 5435Hz.
- *Waveform preprocessing*: notch filtering around 5435Hz with a 20Hz bandwidth.
- *Feature type and feature extraction*: see the various results.
- *Training procedure*: flat start + embedded training with Gaussian splitting:
  1. flat start, i.e. initialization of means and variances of mono-Gaussian pdf models with the mean/variance of the whole training data;
  2. 4 epochs of embedded training;
  3. splitting of the mono-Gaussian pdfs into 3 components mixtures;
  4. 4 epochs of embedded training;
  5. splitting of the 3 Gaussians mixtures pdfs into 5 components mixtures;
  6. 28 epochs of embedded training without further splitting (preliminary studies indicated that this number of training epochs is sufficient to reach convergence).

Table 5.1: Specifications of our speech recognition experiment with the UW-XRMB database.

Compared features

Recognition results comparing the Word Error Rates (WER) related to the Linear Prediction Cepstral Coefficients (LPCC) and to the Log Area Ratios (LAR), using various estimators and various LPC orders, are given in table 5.2 and table 5.3.

In the case of the LPCCs, the 12 cepstral coefficients have been computed by application of the equation (5.1). The required prediction coefficients have been obtained from the reflection coefficients, using the relation (2.3) (and accounting for the zero-valued k_i coefficients in the NUT case).

The Log Area Ratios have been obtained from the application of the relation:

$$\mathrm{lar}_{i+1} = \log\left(\frac{1-k_{i+1}}{1+k_{i+1}}\right) \tag{5.2}$$

The number of LARs in a feature vector is equal to the number of degrees of freedom of the corresponding model.

In the various NUT cases of the type [n/22], the frame-based topology optimization has been applied. Conversely, in the case of the DRM, the topology has been fixed for all the frames to [8/30:3,2,4,6,6,4,2,3.], and an upsampling of the signal from 21.7kHz to 30kHz (using the polyphase method [Bellanger et al., 1976; Elliott, 1987]) has been applied to respect the consistency between the physical model and the DSP model. This was not needed in the other NUT cases, since the 21.7kHz sampling frequency matches approximately the 22 unit sections specification.

No speech recognition experiments have been performed with the ReALiSM method because of the unsolved distortion issues evoked at the end of section 4.3.

An additional result with Mel Filterbank Cepstral Coefficients (MFCC) is given for reference.

Analysis of the results

First, the results check that unconstrained cepstral coefficients issued from a non-parametric or from a parametric spectral analysis (MFCCs and LPCCs) give about the same performance. The choice of the estimator does not appear to play a significant role in the performance of the parametric features.

Next, the results show that the NUT-derived features are competitive when compared to usual cepstral features involving the same number of degrees of freedom.

When the number of DoFs is divided by two, the performance degrades significantly. Apparently, in this case, the higher spectral detail theoretically maintained by the NUT of type [8/22] does not help to keep a low word error rate.

Cepstral Coefficients

12 cepstral coefficients + Energy + Δ + $\Delta\Delta$

Feature type	**DoFs**	**LPC order**	**WER**
MFCC, 24 channels filterbank			4.11%
LPCC Levinson	14	14	4.70%
LPCC Itakura or Burg	14	14	5.45%
NUT LPCC [15/22]	14	21	4.81%
LPCC Levinson	7	7	11.80%
NUT LPCC [8/22]	7	21	15.70%
DRM LPCC [8/30:3,2,4,6,6,4,2,3.]	7	29	19.43%

Table 5.2: Speech recognition results with Cepstral Coefficients.

Log Area Ratios

[DoFs] coefficients + Energy + Δ + $\Delta\Delta$

Feature type	**DoFs**	**LPC order**	**WER**
LAR Burg	14	14	11.48%
LAR Levinson	7	7	13.03%
LAR Itakura or Burg	7	7	13.29%
NUT LAR [8/22]	7	21	17.41%
LAR DRM [8/30:3,2,4,6,6,4,2,3.]	7	29	10.04%

Table 5.3: Speech recognition results with Log Area Ratios.

Results involving the configuration of the DRM model [Mrayati et al., 1988] are given for reference, since their achievement was part of the early objectives of our work. Because of the resampling step, their comparison with the other results is difficult. Nevertheless, the success of the LAR-DRM features is an intriguing point, and the failure of the DRM-LPCC features raises questions as to the "inheritance of relevance" linking the various parameters at various levels of a parametric processing chain. More clearly put, better relative recognition results with Log Area Ratios do not seem to imply better relative results with cepstra, even if the two feature sets derive from the same reflection coefficients.

In any case, the low results obtained with Log Area Ratios reflect the fact that "throwing features into the system" is not sufficient to achieve good recognition results : a finer understanding of the discriminative power of the features *in relation to an appropriate recognition system* is necessary. Speech recognition systems have now reached a degree of sophistication which goes beyond the mere "associative black box" concept. Hence, if cepstral coefficients give good results with HMMs for the very precise reasons developed in the section 5.1.2, a different system, able to draw more relevant discriminant information from the Log Area Ratios, may exist and may still be undiscovered.

Relevance of the results

Indeed, the preliminary results shown here should be taken as a demonstration of the feasibility of using the NUT method in speech recognition, but should by no means be viewed as a definitive assessment of the NUT analysis performances in this framework. As a matter of fact, the following points are still open :

- only one method to derive cepstral coefficients from the NUT-constrained reflection coefficients has been used (namely, the method corresponding to the application of the equation (5.1)). This standard parametric method may not be adapted to the NUT case, and a deeper study of the distortions induced by the NUT models in the cepstral domain is necessary;
- the true advantage of speech production based features should appear with noisy speech, since it is hoped that production-based features will be able to focus on the speech parts of the auditory scene, and that they will remain somewhat "blind" to noise (or at least, more discriminant of the noise and speech parts);
- using the new features as a complement to standard features in a mixed recognition system (multi-stream or other mixture of experts) should enhance the recognition results, by bringing some additional information;
- more extensive testing should be performed on more "standard" speech recognition benchmarking databases, such as Numbers [Cole et al., 1995] or Phonebook [Pitrelli et al., 1995]

for medium vocabulary recognition tasks, or ultimately Switchboard [Linguistic Data Consortium, 1997] if the technique becomes mature enough to be applied to Large Vocabulary Continuous Speech Recognition (LVCSR).

Again, assessing the performance of new features in a pattern recognition framework is indeed a difficult task because their discriminative power depends on the definition of the classes and on the employed discrimination method. Hence, more work is needed to understand precisely under which conditions the inclusion of the NUT constraint or of the ReALiSM constraint within a particular recognition setup can enhance the discriminative power of cepstra used in conjunction with HMMs.

In fact, the class-based structural specialization induced by the NUT paradigm may better express its utility in the framework of Auto-Regressive HMMs [Poritz, 1982; Juang and Rabiner, 1985] (from [Bourlard et al., 1996]). In these models, each state is characterized by an Auto-Regressive signal model (or, equivalently, a Linear Predictor) instead of a multi-Gaussian probability density function, and the likelihoods correspond to the prediction errors of each AR model. Since the use of class-based predictor topologies increases the contrast of the prediction errors between in-class and out-of-class observations, the NUT paradigm may prove useful in enhancing the discriminative power of the AR-HMMs.

In any case, a definitive assessment of the performance of NUT-derived features for pattern recognition may only be conceived as a long-term goal : their use in HMMs represents a sort of sanity check (plus an interesting didactic exercise for the PhD student), but the study of their implications in the development of an alternative and potentially more adapted recognition system exceeds our current scope.

5.2 Speech coding

We have seen little or no use of articulatory paradigms in the speech coding framework. The modern state of the art coding systems seem to rely mainly on perceptual paradigms, or they focus on achieving a graceful degradation in variable bit-rate channels (e.g., for applications in IP-based telephony). However, the reduced dimension and the smoothness of the trajectories brought by articulatory features would make them candidates of choice for an application to low bit-rate coding systems.

As a matter of fact, the analysis systems that we propose achieve a dimensionality reduction with respect to the usual acoustic vectors, and implement the access to feature spaces linked with some speech production paradigms. Lattice filtering being the core of many state of the art coding systems (e.g., the CELP [Schroeder and Atal, 1985], used in GSM telephony), the validation of our extraction method can also take place in the coding domain : it is possible to check if a higher speech quality could be achieved when incorporating the NUT or ReALiSM constraints in the lattice filtering scheme of a coder.

Nevertheless, a careful study of the following points should be performed to assess precisely the interest of our analysis methods in the framework of speech coding :

- *effects of the quantization of the parameters* – the statistics of the constrained parameters extracted from the speech data should be examined to determine a suitable quantization and bit allocation scheme for the parameters of the filter and for the codebook of excitation vectors;
- *listening assessment* – a listening assessment of the waveform rebuilt after parameter quantization and potential degradations should be performed, using the standard Mean Opinion Score (MOS) procedure or some other objective speech quality measures [Quackenbush et al., 1988].

The necessarily limited time span of a thesis has compelled us to leave these points in the domain of future work.

5.3 Speech enhancement : de-noising, production-based speech processing

We have seen in the chapter 3 that the NUT models were able to specialize in the analysis of classes of signal by capturing some of the signal structure in their topology. As a matter of fact, the non-null reflection coefficients occur at the lags corresponding to a maximal correlation between the forward and the backward errors. This capacity could be exploited in a de-noising framework. For instance, using NUT filters trained on clean speech data for the parameterization of noisy speech may allow to increase the robustness of the feature extraction schemes.

As a matter of fact, the structure of the NUT filter codes the places of most significant correlation that are relevant to a class of training signals. If the class-based topologies are reused with noisy instances and without any further optimization, the NUT analysis will keep its focus on the same "places of correlation". This could result in enhanced spectral estimates in noise. To assess the relevance of this idea, the following experiment should be led :

1. train the topologies of some nut filters on some clean instances of a particular class of signals;
2. use the trained topologies or a classical LPC model to estimate the spectral models of the clean instances;
3. add some noise to the analyzed instances;

4. estimate some spectral models of the noisy instances with some classical (unspecialized) LPC models and with the trained NUT models;

5. measure a spectral distortion between the clean spectra (the reference) and the noisy spectra obtained respectively with the classical LPC and the NUT models.

If the noisy spectra obtained with NUTs prove to be less distorted than the noisy spectra in the usual LPC case, it means that the NUT analysis will have somewhat corrected the noisy spectral estimate by focusing on relevant “places of correlation”.

Alternately, in the framework of the NUT model, the spectral warping induced by a voluntary modification of the positions of the non-null reflection coefficients could be exploitable for speaker normalization, e.g. :

1. estimate a global topology for each speaker;

2. estimate speaker-dependent reflection coefficient trajectories, using the speaker-dependent topologies;

3. normalize the features by “forgetting” the original topologies, setting them all to a template configuration (e.g., all ones);

4. re-compute the transfer functions, the spectra and the cepstral features based on the template configuration, to obtain some warped spectral and cepstral features.

After a study of the induced spectral distortions and of the effects on the variability of the spectral and cepstral features, this system could be tested in a speech or speaker recognition framework.

Finally, the spectral estimates issued in relation with production models could be further constrained by acting on the extracted articulatory or pseudo-articulatory trajectories, e.g., through smoothing, thresholding to physiologically plausible values, or notch filtering around human rates of change (see the figure 4.7 on page 81). After the application of these constraints, it is possible to go back to the acoustic features through the computation of some new reflection coefficients inheriting the imposed articulatory constraints. Once again, as a pre-requisite, the induced spectral distortions should be carefully studied.

Chapter 6

Conclusion

From the depths of Digital Signal Processing to the shores of phonetic sciences and articulatory phonology, from the fluid dynamics of lossless tubes to the dry landscapes of C programming, surfing the articles exposing the state of the art methods and applications, this thesis has been the occasion to explore the vast domain of Automatic Speech Processing.

The basic challenge consisted in finding innovative ways of accounting for some speech production paradigms in the feature extraction process. After a study of the available acoustico-articulatory inversion techniques, we have found that the analogy existing between the Linear Prediction (LP) analysis of speech and the lossless acoustic tube models [Markel and Gray, 1976; Wakita, 1979; Bonder, 1983], despite its being part of the DSP tradition for a long time, had not led to recent developments. Yet, it is a fact that most of today's state of the art speech production models (e.g. the Distinctive Regions Model, DRM, [Mrayati et al., 1988] or Maeda's model [Maeda, 1979]) use an acoustic tube as the interface between the acoustic level and the speech production strategy level. Hence, using the LP/tube analogy as a bridge between speech production modeling and DSP appeared to be an innovative and well-founded idea.

Our development of this idea has led to the foundation of two new features extraction methods: the Non-Uniform Topology (NUT) analysis, and the "Relating Acoustics to a Linear Shape Model" (ReALiSM) method.

To establish the NUT model, we have begun with generalizing the traditional lattice filter/lossless tube equivalence to the case of tubes discretized in unequal-length sections. To be able to use these non-uniform tubes as analysis models, we have derived and studied some relevant parametric estimators, based on the observation of the acoustics and on the analytic minimization of a well-defined error criterion. Then, remarking that a fixed non-uniform topology (e.g., the one of the DRM production model) may not be optimal for every part of speech, we have proposed and studied a method to optimize the tube topology from the acoustic observations. To assess this new model, we have studied its spectral modeling properties. They revealed that the NUT estimation is able to achieve a better spectral modeling accuracy than the classical Linear Prediction Coding (LPC) with an equivalent number of parameters. Alternately, we have investigated the potential use of the tube topologies themselves as an analysis tool.

In the case of the ReALiSM model, the first development step has consisted in adapting Maeda's model to the constraints introduced by a connection with the inverse lattice filtering form of LPC. Then, we have integrated the model with the lattice filtering framework through the development of a completely analytic acoustico-articulatory processing chain. Our new system allows to project the solution of the LPC estimation in an articulatory space. Hence, we have provided a qualitative assessment of the LPC solution and of the ReALiSM model in a phonetic framework.

Since no pre-existing software was adapted to the study of these new methods, our research work has involved the development of a complete portable and stand-alone software package, which implements the proposed methods in C language. This software package is freely available for academic purposes as `ftp://ftp.idiap.ch/pub/sacha/Warez/libart.0.0.tar.gz`. The contents and documentation of this package are given in the appendix B.

In addition, Linear Prediction analysis is used as a feature extraction method in most of the main Automatic Speech Processing (ASP) technologies (such as speech coding, speech/speaker recognition, speech synthesis, speech enhancement etc.). Hence, the production-based analysis methods that we have developed create a new gateway for the integration of speech production constraints into the main classical ASP applications. To assess the benefits that they introduce, we have proposed multiple ways to exploit the NUT and ReALiSM systems in various branches of the ASP domain. In particular, we have obtained and discussed encouraging preliminary speech recognition results. Due to the necessarily limited time span of a thesis, we have been obliged to leave the rest of the applicative evaluation for future developments.

Finally, our new systems open a way towards the development of innovative analytic and real-time acoustico-articulatory inversion systems. A more in-depth verification of their true acoustico-articulatory inversion capabilities calls for additional developments in a context involving more focus on some purely phonetic aspects.

Appendix A

The University of Wisconsin speech and X-Ray Microbeam database

A.1 Specifications

For a complete reference document, see [Westbury, 1994].

A.2 Overall characteristics

- *Language*: English
- *Number of speakers*: 48 speakers, 22 males, 26 females
- *Speech volume*:
  - time: approx. 17 minutes per speaker
  - phonemes: approx. 6600 phonemes per speaker with the DARPABET phone set
- *Nature of the utterances*: prompted text,
  - 40% sentences
  - 33% unconnected words and sounds
  - 13% prose (connected speech, long texts)
  - 8% oral motor tasks
  - 6% numbers

A.3 Audio

- *Sampling Freq.*: 21739 Hz
- *Format*: SPHERE, "shorten" type compression
- *Quality*: microphone speech, thin band of machine noise around 5435 Hz, some operator speech during silences, some background white noise.

A.4 Articulation

- *Sampling Freq.*: 146.6 Hz, 6.866 ms period
- *Format*: ASCII, differential coding
- *Quality*: some mistracks marked by a stamp value

A.5 Transcription

- *Orthographic transcription*: prompts given as unformatted ASCII, no time alignment
- *Vocabulary*: whole database → 440 words, unconnected words ("citations" task) → 106 words
- *Phonetic transcription*: none

A.6 Resource management

Database preprocessing

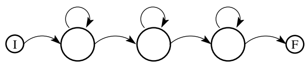
It has been discovered that many audio files were not corresponding to their theoretical orthographic transcription. Inadequacies were of several kinds:

- missing words at the end of the utterances
- cut words at the end of the utterances
- extra words or sentences
- stuttering.

This matching problem has been corrected by implementing an annotation interface suited to the review of the audio and text files, and using it to correct the transcriptions and/or cut adequately the audio files. This correction task has been applied to 75% of the database, comprising the isolated word lists, the sentence lists and the number lists. The remaining 25%, made of long paragraphs, oral motor tasks and sounds, have been considered inappropriate for the training and testing of recognition systems. Leaving these utterances aside brought the advantage of reducing the vocabulary size to 281 words.

Phonetic alignment

A phonetic labelling has been automatically generated through forced alignment of hidden Markov phoneme models. The HMMs have been trained through flat start + embedded training, according to the training procedure described below.

- *Base software platform*: HTK [Young et al., 1999].
- *Phone set*: DARPABET, 39 phonemes (same as DARPA Resource Management phone set), plus beep, silence, preamble and short pause (total 43 models).
- *HMMs type*: monophones, left-right, 3 emitting states. 

- *HMM Emission PDF type*: multi-Gaussians, 5 Gaussian components, diagonal covariances.
- *Training set*: 42 speakers, 51240 words (isolated words 11760, numbers 4116, sentences 35364).
- *Test set*: 6 speakers, 7320 words (isolated words 1680, numbers 588, sentences 5052).
- *Lexicon*: CMU version 0.4.
- *Feature type*: MFCCs + energy + Δ + $\Delta\Delta$.
- *Waveform Preprocessing*: notch filtering around 5435Hz with a 20Hz bandwidth.
- *Training procedure*: flat start + embedded training with Gaussian splitting:
  1. flat start, i.e. initialization of means and variances of mono-Gaussian pdf models with the mean/variance of the whole training data
  2. 4 epochs of embedded training;
  3. splitting of the mono-Gaussian pdfs into 3 components mixtures;
  4. 4 epochs of embedded training;
  5. splitting of the 3 Gaussians mixtures pdfs into 5 components mixtures;

6. 4 epochs of embedded training;
7. additional epochs of embedded training without further splitting, until convergence is reached.

- *Quality of the models*: best word error rates observed for recognition on the test set were 7.8% WER on isolated words and 33.2% WER on the full test set (including continuous speech). These results are average with respect to a pure speech recognition application, but they indicate that the models should be good enough for a forced alignment.
- *Forced alignment assessment*: visual inspection of the forced alignment versus the speech waveform revealed no significant problems.

Speech recognition

See section 5.1.3. Note that the training and test set used in the recognition experiments described in section 5.1.3 differ from those used for the alignment.

Appendix B

Documentation of the experimental software

B.1 Module sigproc

NAME `sigproc.h -- Signal processing functions.`

USAGE

```
#include <sigproc.h>
```

DESCRIPTION This module implements several transformations between common parametric signal modeling features, plus some other useful DSP functions.

AUTHOR *Sacha Krstulović*

B.1.1 Macros

B.1.1.1. Definition of signal processing internal datatype

NAME `Definition of signal processing internal datatype.`

SOURCE

```
#define SigDataType double
```

B.1.1.2. Definition of the error signals of the signal processing library

NAME `Definition of the error signals of the signal processing library.`

SOURCE

```
#define ECORRMETHOD 1 /* Unknown correlation method in autoCorrSeq */
/* (No other error signal yet.) */
```

B.1.2 Functions

B.1.2.1. area_ratio_to_area()

NAME `area_ratio_to_area() -- Turns area ratios into a vector of areas.`

DESCRIPTION For N sections:

$$\begin{cases} S_i = S_{i-1}\, AR_{N-i} \\ \\ S_0 = \text{area of the glottis} \end{cases}$$

SYNOPSIS

```
int area_ratio_to_area( const SigDataType *areaRatio,
                        const int numSections,
                        const SigDataType glottisArea,
                        SigDataType *area );
```

INPUTS

- const SigDataType *areaRatio : the location of a vector of area ratios
- const int numSections : the length of the area and areaRatio vectors
- const SigDataType glottisArea : the glottis area

OUTPUTS

- SigDataType *area : the location of a vector of areas

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `area_to_area_ratio()` [§B.1.2.3], `ki_to_area()` [§B.1.2.9]

B.1.2.2. area_ratio_to_ki()

NAME `area_ratio_to_ki() -- Turns area ratios into reflection coefficients.`

DESCRIPTION

$$k_i = \frac{1 - AR_i}{1 + AR_i}$$

SYNOPSIS

```
int area_ratio_to_ki( const SigDataType *areaRatios, const int numCoeffs,
                      SigDataType *k );
```

INPUTS

- const SigDataType *areaRatios : the location of a vector of area ratios
- const int numCoeffs : the length of the k and areaRatios vectors

OUTPUTS

- SigDataType *k : the target location of a vector of reflection coefficients

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED,always 0.)

SEE ALSO `ki_to_area_ratio()` [§B.1.2.11]

B.1.2.3. area_to_area_ratio()

NAME `area_to_area_ratio() -- Turns areas into area ratios.`

DESCRIPTION For N sections:

$$\begin{cases} \text{area of the glottis} = S_0 \\ \\ AR_{N-i} = \frac{S_i}{S_{i-1}} \end{cases}$$

SYNOPSIS

```
int area_to_area_ratio( const SigDataType *area, const int numSections,
                        SigDataType *areaRatio, SigDataType *glottisArea );
```

INPUTS

- const SigDataType *area : the location of a vector of areas
- const int numSections : the length of the area and areaRatio vectors

OUTPUTS

- SigDataType *areaRatio : the location of a vector of area ratios
- SigDataType *glottisArea : a pointer to the glottis area

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `area_ratio_to_area()` [§B.1.2.1], `area_to_ki()` [§B.1.2.4]

B.1.2.4. area_to_ki()

NAME `area_to_ki() -- Turns areas into reflection coefficients.`

DESCRIPTION For N sections:

$$\begin{cases} \text{area of the glottis} = S_0 \\ \\ k_{N-i} = \frac{1-\frac{S_i}{S_{i-1}}}{1+\frac{S_i}{S_{i-1}}} \end{cases}$$

SYNOPSIS

```
int area_to_ki( const SigDataType *area, const int numCoeffs,
                SigDataType *k, SigDataType *glottisArea );
```

INPUTS

- const SigDataType *area : the location of a vector of areas
- const int numCoeffs : the length of the k vector.

OUTPUTS

- SigDataType *k : the location of a vector of reflection coefficients
- SigDataType *glottisArea : a pointer to the glottis area

Note: the area vector should have one more element than the k vector.

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

NOTES *In this operation, the numbering gets reversed:* k_0 *is the reflection coefficient at the mouth level, while* S_0 *is the area at the glottis.*

SEE ALSO `area_to_ki_no_glottis()` [§B.1.2.5], `ki_to_area()` [§B.1.2.9]

B.1.2.5. area_to_ki_no_glottis()

NAME `area_to_ki_no_glottis() -- Turns areas (without the glottis area) into reflection coefficients.`

DESCRIPTION For N sections:

$$\begin{cases} \text{area of the glottis} = S_0 \\ \\ k_{N-i} = \frac{1-\frac{S_i}{S_{i-1}}}{1+\frac{S_i}{S_{i-1}}} \end{cases}$$

SYNOPSIS

```
int area_to_ki_no_glottis( const SigDataType *area, const int numCoeffs,
                           const SigDataType glottisArea,
                           SigDataType *k );
```

INPUTS

- const SigDataType *area : the location of a vector of areas
- const int numCoeffs : the length of the k vector

OUTPUTS

- SigDataType *k : the location of a vector of reflection coefficients
- SigDataType *glottisArea : a pointer to the glottis area

Note: the area vector has as many elements as the k vector.

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

NOTES *In this operation, the numbering gets reversed:* k_0 *is the reflection coefficient at the mouth level, while* S_0 *is the area after the glottis.*

SEE ALSO `area_to_ki()` [§B.1.2.4], `ki_to_area()` [§B.1.2.9]

B.1.2.6. autocorrelation()

NAME `autocorrelation() -- autocorrelation sequence of a signal window.`

DESCRIPTION [TODO]

B.1.2.7. filter()

NAME `filter() -- Linear filtering`

DESCRIPTION This function applies the difference equation:

$$\begin{aligned} & a_0 * \mathrm{out}[n] = \\ & \quad b_0 \cdot \mathrm{in}[n] + b_1 \cdot \mathrm{in}[n-1] + ... + b_{orderB+1} \cdot \mathrm{in}[n - orderB] \\ & \quad -a_1 \cdot \mathrm{out}[n-1] - ... - a_{orderA+1} \cdot \mathrm{out}[n - orderA] \end{aligned}$$

The filter can be drained if needed:

- if the filetr is not drained, the output vector size is equal to the input vector size;
- conversely, if the filter is drained, the size of the output vector is {length(input) + max(orderA,orderB)}.

BEWARE: in any case, no check is performed as to the length of the output buffer.

If a[0] is not 1.0, the function normalizes the filter coefficients by a[0]. The normalized values are returned in a and b.

SYNOPSIS

```
int filter( const double *in, double *out, const int inLen,
            const double *b, const int orderB,
            const double *a, const int orderA,
            const int flag );
```

INPUTS

- const double *in : location of the input signal buffer of length (const int inLen)
- double *b, const int orderB : FIR filter and its order
- double *a, const int orderA : IIR filter and its order
- const int flag : use DO_DRAIN_FILTER to drain the filter, NO_DRAIN_FILTER otherwise

OUTPUTS

- double *out : the location of the filtered signal buffer
- if a[0] != 1.0, normalized values are returned in a and b

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `ki_to_lpc()` [§B.1.2.13]

B.1.2.8. itakura_saito_dist()

NAME `itakura_saito_dist() -- Itkaura-Saito distortion measure between two spectra`

DESCRIPTION [TODO]

SYNOPSIS

```
double itakura_saito_dist( const SigDataType *s1,
                           const SigDataType *s2,
                           const int numCoeffs );
```

INPUTS

- const SigDataType *s1, const SigDataType *s2 : the location of the two spectra to compare
- const int numCoeffs : the number of spectral coefficients

RETURN VALUE a double indicating the Itakura-Saito distance between s1 and s2.

SEE ALSO `lpc_to_spectrum()` [§B.1.2.20]

B.1.2.9. ki_to_area()

NAME `ki_to_area() -- Turns reflection coefficients into areas.`

DESCRIPTION For N sections:

$$\begin{cases} S_i = S_{i-1}\frac{1-k_{N-i}}{1+k_{N-i}} \\ \\ S_0 = \text{area of the glottis} \end{cases}$$

SYNOPSIS

```
int ki_to_area( const SigDataType *k, const int numCoeffs,
                const SigDataType glottisArea,
                SigDataType *area );
```

INPUTS

- const SigDataType *k : the location of a vector of reflection coefficients
- const int numCoeffs : the length of the k vector.
- const SigDataType glottisArea : the glottis area

OUTPUTS

- SigDataType *areas : the target location of a vector of areas

Note: The area vector should have one more element than the k vector.

RETURN VALUE 0 if everything OK, <0 otherwise. (UNUSED, always 0.)

NOTES *In this operation, the numbering gets reversed: k_0 is the reflection coefficient at the mouth level, while S_0 is the area at the glottis.*

SEE ALSO `ki_to_area_no_glottis()` [§B.1.2.10], `ki_to_area_ratio()` [§B.1.2.11], `area_to_ki()` [§B.1.2.4]

B.1.2.10. ki_to_area_no_glottis()

NAME `ki_to_area_no_glottis() -- Turns reflection coefficients into areas, but does not include the glottis area in the resulting area vector.`

DESCRIPTION For N sections:

$$\begin{cases} S_i = S_{i-1}\frac{1-k_{N-i}}{1+k_{N-i}} \\ \\ S_0 = \text{area of the glottis} \end{cases}$$

SYNOPSIS

```
int ki_to_area_no_glottis( const SigDataType *k, const int numCoeffs,
                           const SigDataType glottisArea,
                           SigDataType *area );
```

INPUTS

- const SigDataType *k : the location of a vector of reflection coefficients
- const int numCoeffs : the length of the k vector.
- const SigDataType glottisArea : the glottis area

OUTPUTS

- SigDataType *areas : the target location of a vector of areas

Note: the area vector has as many elements as the k vector.

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

NOTES *In this operation, the numbering gets reversed:* k_0 *is the reflection coefficient at the mouth level, while* S_0 *is the area after the glottis.*

SEE ALSO `ki_to_area()` [§B.1.2.9], `ki_to_area_ratio()` [§B.1.2.11], `area_to_ki()` [§B.1.2.4]

B.1.2.11. ki_to_area_ratio()

NAME `ki_to_area_ratio() -- Turns reflection coefficients into area ratios.`

DESCRIPTION

$$AR_i = \frac{1 - k_i}{1 + k_i}$$

SYNOPSIS

```
int ki_to_area_ratio( const SigDataType *k, const int numCoeffs,
                      SigDataType *areaRatios );
```

INPUTS

- const SigDataType *k : the location of a vector of reflection coefficients
- const int numCoeffs : the length of the k and areaRatios vectors

OUTPUTS

- SigDataType *areaRatios : the target location of a vector of area ratios

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `ki_to_log_area_ratio()` [§B.1.2.12], `area_ratio_to_ki()` [§B.1.2.2]

B.1.2.12. ki_to_log_area_ratio()

NAME `ki_to_log_area_ratio() -- Turns reflection coefficients into log area ratios.`

DESCRIPTION

$$LAR_i = \log\left(\frac{1 - k_i}{1 + k_i}\right)$$

SYNOPSIS

```
int ki_to_log_area_ratio( const SigDataType *k, const int numCoeffs,
                          SigDataType *logAreaRatios );
```

INPUTS

- const SigDataType *k : the location of a vector of reflection coefficients
- const int numCoeffs : the length of the k and areaRatios vectors

OUTPUTS

- SigDataType *logAreaRatios : the target location of a vector of log area ratios

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED,always 0.)

SEE ALSO `ki_to_area_ratio() [§B.1.2.11], log_area_ratio_to_ki() [§B.1.2.15]`

B.1.2.13. ki_to_lpc()

NAME `ki_to_lpc() -- Turns reflection coefficients into Linear Prediction Coefficients.`

DESCRIPTION For m between 0 and lpcOrder:

$$\begin{cases} a_i^{(m+1)} = a_i^{(m)} - k_{m+1} a_{m+1-i}^{(m)} \\ a_0^{(0)} = 1 \\ a_i^{(0)} = 0, \ \ \forall i \neq 0 \end{cases}$$

or equivalently, at each step m:

$$\begin{cases} a_0^{(m+1)} = a_0^{(m)} \\ a_1^{(m+1)} = a_1^{(m)} - k_{m+1} a_m^{(m)} \\ \vdots \\ a_m^{(m+1)} = a_m^{(m)} - k_{m+1} a_1^{(m)} \\ a_{m+1}^{(m+1)} = -k_{m+1} a_0^{(m)} \end{cases}$$

SYNOPSIS

```
int ki_to_lpc( const SigDataType *k, const int lpcOrder,
               SigDataType *a );
```

INPUTS

- const SigDataType *k : the location of a vector of reflection coefficients
- const int lpcOrder : the length of the k vector

OUTPUTS

- SigDataType *a : the location of a vector of LP coefficients

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `ki_to_lpc_no_a0() [§B.1.2.14], lpc_to_ki() [§B.1.2.16], ki_to_area_ratio() [§B.1.2.11], ki_to_log_area_ratio() [§B.1.2.12].`

B.1.2.14. ki_to_lpc_no_a0()

NAME `ki_to_lpc_no_a0() -- Turns reflection coefficients into Linear Prediction Coefficients, ignoring the default $a_0 = 1$.`

DESCRIPTION For m between 0 and lpcOrder:

$$\begin{cases} a_i^{(m+1)} = a_i^{(m)} - k_{m+1} a_{m+1-i}^{(m)} \\ a_0^{(0)} = 1 \\ a_i^{(0)} = 0, \ \ \forall i \neq 0 \end{cases}$$

or equivalently, at each step m:

$$\begin{cases} a_0^{(m+1)} = a_0^{(m)} \\ a_1^{(m+1)} = a_1^{(m)} - k_{m+1} a_m^{(m)} \\ \vdots \\ a_m^{(m+1)} = a_m^{(m)} - k_{m+1} a_1^{(m)} \\ a_{m+1}^{(m+1)} = -k_{m+1} a_0^{(m)} \end{cases}$$

SYNOPSIS

```
int ki_to_lpc_no_a0( const SigDataType *k, const int lpcOrder,
                     SigDataType *a );
```

INPUTS

- const SigDataType *k : the location of a vector of reflection coefficients
- const int lpcOrder : the length of the k and lpc vectors

OUTPUTS

- SigDataType *a : the location of a vector of LP coefficients

RETURN VALUE 0 if everything OK, <0 otherwise. (UNUSED, always 0.)

SEE ALSO `ki_to_lpc()` [§`B.1.2.13`], `lpc_to_ki()` [§`B.1.2.16`], `ki_to_area_ratio()` [§`B.1.2.11`], `ki_to_log_area_ratio()` [§`B.1.2.12`].

B.1.2.15. log_area_ratio_to_ki()

NAME `log_area_ratio_to_ki() -- Turns log area ratios into reflection coefficients.`

DESCRIPTION

$$k_i = \frac{1 - e^{LAR_i}}{1 + e^{LAR_i}}$$

SYNOPSIS

```
int log_area_ratio_to_ki( const SigDataType *logAreaRatios,
                          const int numCoeffs,
                          SigDataType *k );
```

INPUTS

- const SigDataType *logAreaRatios : the location of a vector of area ratios
- const int numCoeffs : the length of the k and areaRatios vectors

OUTPUTS

- SigDataType *k : the target location of a vector of reflection coefficients

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `ki_to_log_area_ratio()` [§B.1.2.12]

B.1.2.16. lpc_to_ki()

NAME `lpc_to_ki() -- Turns Linear Prediction Coefficients into reflection coefficients.`

DESCRIPTION [TODO]

SYNOPSIS

```
int lpc_to_ki( SigDataType *a, const int lpcOrder,
               SigDataType *k );
```

INPUTS

- SigDataType *a : the location of a vector of *(lpcOrder+1)* LP coefficients
- const int lpcOrder : the length of the k vector

OUTPUTS

- SigDataType *k : the location of a vector of reflection coefficients

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `ki_to_lpc()` [§B.1.2.13], `ki_to_lpc_no_a0()` [§B.1.2.14]

B.1.2.17. lpc_to_log_spec()

NAME `lpc_to_log_spec() -- Computation of the log magnitude of a complex spectrum from all-pole LPC coefficients.`

DESCRIPTION Log magnitude of the evaluation of the transfer function on the unit circle.

SYNOPSIS

```
int lpc_to_log_spec(const SigDataType *a, const int order,
                    SigDataType *omega, const int omegaLen,
                    SigDataType *spectrum );
```

INPUTS

- const SigDataType *a : the denominator's polynomial coefficients, *including leading a[0] = 1*
- const int order : the order of the denominator (i.e. number of a coefficients)
- const SigDataType *omega : the location of a vector of normalized frequencies (in radians/sample, usually between $-\pi$ and pi).
- const int omegaLen : the length of the frequency vector

OUTPUTS

- Cmplx *spectrum : the location of the log magnitude of the resulting complex spectrum

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `lpc_to_spectrum()` [§B.1.2.20], `lpc_to_mag_spec()` [§B.1.2.17], `c_mag()` [§B.2.3.3], `c_phase` [§B.2.3.3]

B.1.2.18. lpc_to_lpcc()

NAME `lpc_to_lpcc() -- Computation of LPC-derived cepstral coefficients`

DESCRIPTION

$$c_n = -a_n - \frac{1}{n}\sum_{i=1}^{n-1}(n-i)a_i c_{n-i} \quad \text{for} \ \ n > 0$$

where c_n are the cepstral coefficients and a_i are the linear prediction coefficients with $a_0 = 1$ and $a_i = 0$ for $i > n$. WARNING: c_0 (which should be the log of the spectral power) is not computed in this function. Fill it by hand.

SYNOPSIS

```
int lpc_to_lpcc( const SigDataType*a, const int lpcOrder,
                 SigDataType *c, const int nCep );
```

INPUTS

- const SigDataType *a : the lpc coefficients, *including the leading a[0] = 1*
- const int lpcOrder : the number of LPC coefficients
- const int nCep : the number of output cepstral coefficients

OUTPUTS

- SigDataType *c : the location of the resulting vector of cepstral coefficients

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `lpc_to_spectrum()` [§B.1.2.20], `lpc_to_log_spec()` [§B.1.2.17], `lpc_to_mag_spec()` [§B.1.2.19], `c_mag()` [§B.2.3.3], `c_phase` [§B.2.3.3]

B.1.2.19. lpc_to_mag_spec()

NAME `lpc_to_mag_spec() -- Computation of the magnitude of a complex spectrum from all-pole LPC coefficients.`

DESCRIPTION Magnitude of the evaluation of the transfer function on the unit circle.

SYNOPSIS

```
int lpc_to_mag_spec( const SigDataType *a, const int order,
                     const SigDataType *omega, const int omegaLen,
                     SigDataType *spectrum );
```

INPUTS

- const SigDataType *a : the denominator's polynomial coefficients, *including the leading a[0] = 1*
- const int order : the order of the denominator (i.e. number of a coefficients)
- const SigDataType *omega : the location of a vector of normalized frequencies (in radians/sample, usually between $-\pi$ and pi).
- const int omegaLen : the length of the frequency vector

OUTPUTS

- Cmplx *spectrum : the location of the magnitude of the resulting complex spectrum

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `lpc_to_spectrum()` [§B.1.2.20], `lpc_to_log_spec()` [§B.1.2.17], `c_mag()` [§B.2.3.3], `c_phase` [§B.2.3.3]

B.1.2.20. lpc_to_spectrum()

NAME `lpc_to_spectrum() -- Computation of a complex spectrum from all-pole LPC coefficients.`

DESCRIPTION Evaluation of the transfer function on the unit circle.

SYNOPSIS

```
int lpc_to_spectrum(const SigDataType *a, const int order,
                    const SigDataType *omega, const int omegaLen,
                    Cmplx *spectrum );
```

INPUTS

- const SigDataType *a : the denominator's polynomial coefficients, *including the leading a[0] = 1*
- const int order : the order of the denominator (i.e. number of a coefficients)
- const SigDataType *omega : the location of a vector of normalized frequencies (in radians/sample, usually between $-\pi$ and pi).
- const int omegaLen : the length of the frequency vector

OUTPUTS

- Cmplx *spectrum : the location of the resulting complex spectrum

RETURN VALUE 0 if everything OK, < 0 otherwise. (UNUSED, always 0.)

SEE ALSO `lpc_to_mag_spec()` [§B.1.2.19], `lpc_to_log_spec()` [§B.1.2.17], module `cmplxmth.h` [§B.2], `c_mag()` [§B.2.3.3], `c_phase` [§B.2.3.3]

B.1.2.21. toeplitz()

NAME `toeplitz() -- make a toeplitz matrix from a real sequence.`

DESCRIPTION [TODO]

B.2 Module cmplxmth

NAME `cmplxmth.h -- Implementation of standard complex math functions.`

USAGE

```
#include <cmplxmth.h>
```

AUTHOR *Sacha Krstulović*

B.2.1 Structures

B.2.1.1. Definition of complex datatype in cartesian form

NAME `Definition of complex datatype in cartesian form.`

SOURCE

```
typedef struct {
    CmplxDataType re;
    CmplxDataType im;
} CartesianCmplx;

/* Cartesian is the default complex type. */
#define Cmplx CartesianCmplx
```

B.2.1.2. Definition of complex datatype in polar form

NAME `Definition of complex datatype in polar form.`

SOURCE

```
typedef struct {
    CmplxDataType mag;
    CmplxDataType phase;
} PolarCmplx;
```

B.2.2 Macros

B.2.2.1. Definition of complex numbers' internal datatype

NAME `Definition of complex numbers' internal datatype.`

SOURCE

```
#define CmplxDataType double
```

B.2.3 Functions

B.2.3.1. Transformations between cartesian and polar complex

NAME Transformations between cartesian and polar complex representations.

SOURCE

```
    int cart2pol( Cmplx *inCart, const unsigned int len, PolarCmplx *outPol);
    int cart2pow( Cmplx *inCart, const unsigned int len, CmplxDataType *outPow);
    int pol2cart( PolarCmplx *inPol, const unsigned int len, Cmplx *outCart);
```

B.2.3.2. Standard algebraic functions for complex type

NAME Standard algebraic functions for complex type.

SOURCE

```
    CmplxDataType c_real(const Cmplx x); /* real part */
    CmplxDataType c_imag(const Cmplx x); /* imaginary part */
    CmplxDataType c_abs (const Cmplx x); /* magnitude */
#define c_mag c_abs

    CmplxDataType c_arg (const Cmplx x); /* phase */
#define c_phase c_arg

    Cmplx make_cmplx(const  CmplxDataType re ,const  CmplxDataType im );
    Cmplx c_pure_i( void );
#define c_i c_pure_i()

    Cmplx c_add(const Cmplx a, const Cmplx b ); /* a + b */
    Cmplx c_sub(const Cmplx a, const Cmplx b ); /*  a - b  */
    Cmplx c_scl(const CmplxDataType x, const Cmplx a ); /*  x*a  */
    Cmplx c_mul(const Cmplx a, const Cmplx b ); /*  a*b  */
    Cmplx c_div(const Cmplx a, const Cmplx b ); /*  a/b  */
    Cmplx c_add_mul(const Cmplx a, const Cmplx b, const Cmplx c ); /* a + b*c */
    Cmplx c_sub_mul(const Cmplx a, const Cmplx b, const Cmplx c ); /* a - b*c */
    Cmplx c_inv(const Cmplx a ); /* 1/a */
    Cmplx c_sqr(const Cmplx a ); /* a^2 */
```

B.2.3.3. Standard math functions adapted to complex type

NAME Standard math functions adapted to complex type.

SOURCE

```
    Cmplx c_cos (const Cmplx x);
    Cmplx c_cosh (const Cmplx x);
    Cmplx c_sin (const Cmplx x);
    Cmplx c_sinh (const Cmplx x);
    Cmplx c_log (const Cmplx x);
    Cmplx c_sqrt (const Cmplx x);
```

B.3 Module libart

NAME `libart -- library for articulatory modeling.`

USAGE

```
#include <libart.h>
```

DESCRIPTION This library gives functions useful for the implementation of profile-based articulatory models (e.g. Maeda's model).

It is a (deep) mutation of Maeda's original vlam_lib source code.

NOTES *Typographic conventions:*

- *Variables: myVarToto*
- *Types: MyType*
- *Functions: my_function*
- *Constants: MY_CSTE*

AUTHOR *Sacha Krstulović*

B.3.1 Structures

B.3.1.1. AlphaBetaMap

NAME `AlphaBetaMap -- Structure for the coefficients of the alpha-beta transform.`

USAGE

```
AlphaBetaMap myABMap;
AlphaBetaMap *myABMapPtr;
```

DESCRIPTION See source code.

SEE ALSO `default_abmap()` [§B.3.4.7], `init_abmap()` [§B.3.4.11], `release_abmap()` [§B.3.4.19].

SOURCE

```
typedef struct{
    int numCoeffs;  /* Pretty self explanatory. */
    double *alpha;
    double *beta;
} AlphaBetaMap;
```

B.3.1.2. AreaFunction

NAME `AreaFunction -- Structure for storing area functions.`

USAGE

```
AreaFunction myAreaFun;
AreaFunction *myAreaFunPtr;
```

DESCRIPTION See source code.

SEE ALSO `init_area_function() [§B.3.4.12], release_area_function() [§B.3.4.20].`

SOURCE

```
typedef struct{
    int numSections;  /* Number of elementary tubes */
    double *A;        /* Area of tube sections */
    double *x;        /* Length of tube sections */
} AreaFunction;
```

B.3.1.3. Contour

NAME `Contour -- Structure for storing the vocal tract contour.`

USAGE

```
Contour myContour;
Contour *myContourPtr;
```

DESCRIPTION This structure holds the x and y coordinates of the vocal tract contour points. It makes a distinction between the interior contour (i.e. the tongue side) and the exterior contour (i.e. the hard palate side).

NOTES *All positions are measured in centimeters. Hence, measures made on the contour should be proportional to "human reality".*

SEE ALSO `init_contour() [§B.3.4.13], release_contour() [§B.3.4.21].`

SOURCE

```
typedef struct{
    int numSections;  /* Number of contour sections */
    double *diameter; /* Distance between anterior and posterior edge along
                         one section */
    double *ivtX;     /* Interior edge */
    double *ivtY;
    double *evtX;     /* Exterior edge */
    double *evtY;
    double lipH;      /* Lips height for frontal view */
    double lipW;      /* Lips width for frontal view */
} Contour;
```

B.3.1.4. Model

NAME `Model -- Structure for storing the model factors and the model shape.`

USAGE

```
Model myModel;
Model *myModelPtr;
```

DESCRIPTION This structure holds any information pertaining to the composition of profile shapes by linear combination.

The shapes are made of three sections (larynx, tongue and lips) which are measured in different ways:

- for the larynx, the measures represent the position of the two lowest larynx points
- for the tongue, the measures represent the abscissa of the tongue profile *along the semipolar gridlines*
- for the lips, the measures represent the aperture, the protrusion and the position of a reference point
- the wall position is also measured along the semipolar gridlines.

These measures will be later transformed into a set of 2D points forming a contour by projections adapted to the three different zones (larynx, tongue and lips).

SEE ALSO `default_model()` [§B.3.4.10], `release_model()` [§B.3.4.23], `default_from_file()` [§B.3.4.8].

SOURCE

```
typedef struct{
    /* LRX */
    int numFactLrx;   /* Number of factors for making Larynx shapes */
    int dimLrx;       /* Dimension of larynx piece of shape */
    double *meanLrx;  /* Mean of larynx factor */
    double *stdevLrx; /* Standard deviation of larynx factor */
    double **ALrx;    /* Factor shapes */
    double *vLrx;     /* Vector for storing shapes after linear combination of
                 factors */
               /* See libart.c for further comments */
    /* TNG */
    int numFactTng;   /* Same as above, for the tongue-related piece of shape */
    int dimTng;
    double *meanTng;
    double *stdevTng;
    double **ATng;
    double *vTng;
    /* LIP */
    int numFactLip;  /* Same as above, for the lips-related piece of shape */
    int dimLip;
    double *meanLip;
    double *stdevLip;
    double **ALip;
    double *vLip;
    /* WAL */
    int numFactWal;  /* Same as above, for the wall-related piece of shape */
    int dimWal;
    double *meanWal;
    double *stdevWal;
    double **AWal;
    /* There is no {double *vWal} because the wall shape will always be
       considered as fixed to its mean value. */
} Model;
```

B.3.1.5. SemipolGrid

NAME `SemipolGrid -- Structure for storing the semipolar grid.`

USAGE

```
Semipolgrid myGrid;
Semipolgrid  *myGridPtr;
```

DESCRIPTION This structure holds any information pertaining to the coordinates used to measure the factor shapes or necessary to project them onto the 2D plane. In particular, it defines the semipolar grid and the position of the larynx and lips with respect to the grid.

SEE ALSO `default_grid()` [§B.3.4.9], `update_grid()` [§B.3.4.25], `release_grid()` [§B.3.4.22], `default_from_file()` [§B.3.4.8].

SOURCE

```
typedef struct{
    /* Geometric specs */
    double dl;          /* Gridline spacing in linear regions (cm) */
    double r;           /* Gridline length (cm) */
    double omegaDeg;    /* Tilt of the grid with respect to the vertical, in
                 degrees */
    double thetaDeg;    /* Angular increment in the polar zone, in degrees. */
    double xOrigin;     /* X position of the grid's origin, in centimeters.
                 The grid's origin is where the lines meet in the
                 polar region of the grid. */
    double yOrigin;     /* X position of the grid's origin, in centimeters. */
    double pharynxScale; /* Scaling factor for the pharynx region, used when
                 adapting the original model to a new locutor. */
    double mouthScale;  /* Scaling factor for the mouth region, used when
                 adapting the original model to a new locutor. */
    /* Gridlines specs */
    int numLinesBack;  /* Number of lines in the back (larynx) region of the
                grid */
    int numLinesPol;   /* Number of lines in the polar region of the grid */
    int numLinesFront; /* Number of lines in the front (mouth) region of the
                grid */
    int numLinesTot;   /* Total number of lines */
    /* Lines data in 2D plane */
    double *inteX;      /* X-Y Coordinates of the extremal points of the
                gridlines in the 2D plane. There is an interior
                point (tongue side), and an exterior point (hard
                palate side), hence the inte/exte distinction. */
    double *inteY;
    double *exteX;
    double *exteY;
    /* Additional line data */
    double *angle;      /* Angle of each line */
    double *sinAngle;   /* Cosine of each line's angle,
                 useful for projections */
    double *cosAngle;   /* Sine of each line's angle,
                 useful for projections */
    /* Lips position specification */
    double inciX;       /* X-Y position of upper incisor, useful for locating
                 the lip tube. */
    double inciY;
} SemipolGrid;
```

B.3.2 Constants

B.3.2.1. INCI_LIP

NAME INCI_LIP -- Constant.

USAGE

Used in the contour_to_area_orig() <REF "contour_to_area_orig()"> function.

DESCRIPTION Correction of distance between incisor and upper lip in projected model, in centimeters. This sets the position of the lips in the contour shape.

SOURCE

```
#define INCI_LIP 0.8
```

B.3.2.2. PI

NAME PI -- Constant π for trigonometric operations.

SOURCE

```
#define PI 3.14159265359
```

B.3.2.3. SIZE_CORRECTION

NAME SIZE_CORRECTION -- Constant.

USAGE

```
Used in the contour_to_area_orig() <REF "contour_to_area_orig()">
function.
```

DESCRIPTION 10% size increase (presumably to go from model extracted from a female subject to a model that has a male voice) applied in the contour to area transformation.

SOURCE

```
#define SIZE_CORRECTION 1.10
```

B.3.2.4. TWO_PI

NAME TWO_PI -- Constant 2π for trigonometric operations.

SOURCE

```
#define TWO_PI 6.28318530718
```

B.3.3 Macros

B.3.3.1. deg2rad, rad2deg

NAME deg2rad, rad2deg -- Degrees/radians conversions.

SOURCE

```
#define deg2rad(A) ( A * PI/180 )
#define rad2deg(A) ( A * 180/PI )
```

B.3.3.2. dist

NAME dist -- Distance between two points.

SOURCE

```
#define dist(AX,AY,BX,BY) \
     ( sqrt( \
        ( (double)(AX) - (double)(BX) ) * ( (double)(AX) - (double)(BX) ) \
      + ( (double)(AY) - (double)(BY) ) * ( (double)(AY) - (double)(BY) ) \
     ) )
```

B.3.3.3. max, min

NAME `max, min -- Minimum and maximum of two numbers`

SOURCE

```
#define max(A,B) ((A) > (B) ? (A) : (B))
#define min(A,B) ((A) < (B) ? (A) : (B))
```

B.3.4 Functions

B.3.4.1. area_to_contour_cylmean()

NAME `area_to_contour_cylmean() -- Turns an area function into a contour, cylinders mean method.`

DESCRIPTION Postulates cylindric sections and interfaces = mean diameter between adjacent sections.

SYNOPSIS

```
int area_to_contour_cylmean( const AreaFunction af,
                             const AlphaBetaMap abMap, const int abMapSync,
                             const SemipolGrid grid, const int gridOffset,
                             const Model model, const int modelSync,
                             const double glottisArea,
                             Contour *vtContour )
```

INPUTS

- const AreaFunction af : an area function
- const AlphaBetaMap abMap : the alpha-beta map to use
- const int abMapSync : the shift between the alpha-beta map indexes and the contour indexes
- const SemipolGrid grid : the semipolar grid where the operation is performed
- const int gridOffset : the shift between the grid indexes and the contour indexes
- const Model model : the Model structure holding the back wall contour
- const int modelSync : the shift between the model indexes and the contour indexes
- const double glottisArea : the area of the (fixed) glottis section.
- Contour *vtContour : a previously allocated Contour object

OUTPUTS

- Contour *vtContour : a filled Contour object

RETURN VALUE 0 if everything OK, < 0 otherwise.

SEE ALSO `area_to_contour_interface()` [§B.3.4.2].

B.3.4.2. area_to_contour_interface()

NAME `area_to_contour_interface() -- Turns an area function into a contour, interfaces method.`

DESCRIPTION Uses inverse alpha-beta transform on interfaces (gridlines) only.

SYNOPSIS

```
int area_to_contour_interface( const AreaFunction af,
                               const AlphaBetaMap abMap, const int abMapSync,
                               const SemipolGrid grid, const int gridOffset,
                               const Model model, const int modelSync,
                               const double glottisArea,
                               Contour *vtContour );
```

INPUTS

- const AreaFunction af : an area function
- const AlphaBetaMap abMap : the alpha-beta map to use
- const int abMapSync : the shift between the alpha-beta map indexes and the contour indexes
- const SemipolGrid grid : the semipolar grid where the operation is performed
- const int gridOffset : the shift between the grid indexes and the contour indexes
- const Model model : the Model structure holding the back wall contour
- const int modelSync : the shift between the model indexes and the contour indexes
- const double glottisArea : the area of the (fixed) glottis section (unused in this function; kept for compatibility with area_to_contour_cylmean() [§ B.3.4.1]).
- Contour *vtContour : a previously allocated Contour object

OUTPUTS

- Contour *vtContour : a filled Contour object

RETURN VALUE 0 if everything OK, < 0 otherwise.

SEE ALSO `area_to_contour_cylmean()` [§ B.3.4.1].

B.3.4.3. areafun_to_mag_spectrum()

NAME `areafun_to_mag_spectrum() -- Computation of the magnitude of the spectrum from the area function of a given lossy tube.`

DESCRIPTION [TODO]

REFERENCES ATAL 1978 and BADIN 84.

SYNOPSIS

```
int areafun_to_mag_spectrum( const double *freq, const int freqLen,
                             const double sampleFreq,
                             const double *area, const double *length,
                             const int numSections,
                             double *spec );
```

INPUTS

- const double *freq : a normalized frequency vector (in radians)
- const int freqLen : the length of the frequency vector
- const double sampleFreq : sampling frequency (in Hz)
- const double area : an area function (area of each tube sections)
- const double length : length of each tube section
- const int numSections : number of tube sections
- double *spec : a vector able to hold "freqLen" values of type double

OUTPUTS

- double *spec : the magnitude of the resulting complex spectrum

RETURN VALUE 0 if everything OK, < 0 otherwise.

B.3.4.4. areafun_to_spectrum()

NAME `areafun_to_spectrum() -- Computation of a spectrum from the area function of a given lossy tube.`

DESCRIPTION [TODO]

REFERENCES ATAL 1978 and BADIN 84.

SYNOPSIS

```
int areafun_to_spectrum(const Cmplx *freq, const int freqLen,
                        const AreaFunction tube,
                        Cmplx *spectrum );
```

INPUTS

- const Cmplx *freq : a complex frequency vector (in Hz)
- const int freqLen : the length of the frequency vector
- const AreaFunction tube : an area function
- Cmplx *spectrum : a previously allocated complex vector

OUTPUTS

- Cmplx *spectrum : the resulting complex spectrum

RETURN VALUE 0 if everything OK, < 0 otherwise.

B.3.4.5. compose_shape()

NAME `compose_shape() -- Compute vectorial representation of the articulator positions for given parameter values.`

DESCRIPTION Compose_shape() operates the linear combination of elementary factor shapes weighted by articulatory parameter values.

Articulatory parameters are defined as follows in a 7 elements vector:

- pa[0] : jaw
- pa[1] : tongue-body position
- pa[2] : tongue-body shape
- pa[3] : tongue-tip position
- pa[4] : lip height (aperture)
- pa[5] : lip protrusion
- pa[6] : larynx height

SYNOPSIS

```
int compose_shape( const double *pa, Model model );
```

INPUTS

- const double *pa : the array of seven articulatory parameters.
- Model model : a Model [§ B.3.1.4] structure previously initialized with elementary factor shapes, using default_model() [§ B.3.4.10] or default_from_file() [§ B.3.4.8].

OUTPUTS

- Model model : holds the shape resulting from linear combination of factors.

RETURN VALUE 0 if everything OK, < 0 otherwise.

B.3.4.6. contour_to_area()

NAME `contour_to_area() -- Turns a profile into an area function.`

DESCRIPTION This function exists in three version:

- contour_to_area_orig : Maeda's original version, with the weird "magic" transformation of the surfaces;
- contour_to_area_interfaces : a version acting only on a cylindric transformation from the interfaces;
- contour_to_area_fixlen : same as the latter, but in a fixed length version.

INPUTS

- const Contour vtContour : a vocal tract contour
- const AlphaBetaMap abMap : the alpha-beta transform coefficients
- AreaFunction af : a previously allocated area function

OUTPUTS

- AreaFunction af : a filled area function

RETURN VALUE 0 if everything OK, < 0 otherwise.

SOURCE

```
/* Maeda's original version with conservation of profile area */
int contour_to_area_orig( const Contour vtContour, const AlphaBetaMap abMap,
                  AreaFunction *af );
/* Version acting only on the interfaces */
int contour_to_area_interface( const Contour vtContour,
                        const AlphaBetaMap abMap,
                        const int abMapOffset,
                        AreaFunction *af );
int contour_to_area_fixlen( const Contour vtContour,
                     const AlphaBetaMap abMap,
                     AreaFunction *af );
```

B.3.4.7. default_abmap()

NAME `default_abmap() -- initializes an alpha-beta map object from values hardcoded in lib_def.h.`

DESCRIPTION The hardcoded default values correspond to Maeda's original values.

INPUTS void

RETURN VALUE A new alpha-beta map object initialized to Maeda's original default values.

SEE ALSO `default_from_file()` [§B.3.4.8], `null_abmap()` [§B.3.4.14], `init_abmap()` [§B.3.4.11], `release_abmap()` [§B.3.4.19].

B.3.4.8. default_from_file()

NAME `default_from_file() -- Read the model specifications in a file, and initializes all important objects.`

SYNOPSIS

```
int default_from_file( const char *fileName,
                       SemipolGrid *grid,
                       AlphaBetaMap *abMap,
                       Model *model,
                       double pharynxScale,
                       double mouthScale );
```

INPUTS

- const char *fileName : the name of the file holding the model and grid specifications
- double pharynxScale, larynxScale : the scaling factors (unspecified in the file)
- ... : pointers on allocated/uninitialized semipolar grid, alpha-beta map and model structures.

OUPUTS

- Initialized semipolar grid, alpha-beta map and model structures.

RETURN VALUE 0 if everything OK, < 0 otherwise.

B.3.4.9. default_grid()

NAME `default_grid() -- initializes a grid object from values hardcoded in lib_def.h.`

DESCRIPTION [TODO]

SYNOPSIS

```
SemipolGrid default_grid( void );
```

RETURN VALUE A new grid object initialized to Maeda's original default values.

NOTES *The hardcoded default values correspond to Maeda's original values.*

SEE ALSO `init_from_file() [§??], null_grid() [§B.3.4.17], update_grid() [§B.3.4.25], release_grid() [§B.3.4.22].`

B.3.4.10. default_model()

NAME `default_model() -- initializes a model object from values hardcoded in lib_def.h.`

DESCRIPTION The hardcoded default values correspond to Maeda's original values.

SYNOPSIS

```
Model default_model( void );
```

INPUTS void

RETURN VALUE A new model object initialized to Maeda's original default values.

SEE ALSO `default_from_file() [§B.3.4.8], null_model() [§B.3.4.18], release_model() [§B.3.4.23].`

B.3.4.11. init_abmap()

NAME `init_abmap() -- allocates an alpha-beta map object.`

DESCRIPTION returns an alpha-beta map object able to hold numCoeffs coefficient pairs (necessary memory is allocated).

INPUTS

- int numCoeffs : number of alpha-beta coefficients pairs

RETURN VALUE A new alpha-beta map object with all alpha and beta values set to zero.

SEE ALSO `default_from_file() [§B.3.4.8], default_abmap() [§B.3.4.7], null_abmap() [§B.3.4.14], release_abmap() [§B.3.4.19].`

B.3.4.12. init_area_function()

NAME `init_area_function() -- allocates an area function object.`

DESCRIPTION Returns an area function object able to hold numSections sections (necessary memory is allocated).

INPUTS

- int numSections : number of sections in the area function

RETURN VALUE A new AreaFunction object with all A and x values set to zero.

SEE ALSO `null_area_function() [§B.3.4.15], release_area_function() [§B.3.4.20].`

B.3.4.13. init_contour()

NAME `init_contour() -- returns a contour object able to hold numSections sections.`

DESCRIPTION Necessary memory is allocated.

SYNOPSIS

```
Contour init_contour( int numSections );
```

INPUTS

- int numSections : number of sections (or internal/external contour points *pairs*) in the contour

RETURN VALUE A new Contour object with all evtX, evtY, ivtX and ivtY values set to zero or NULL if failed.

SEE ALSO `null_contour() [§B.3.4.16], release_contour() [§B.3.4.21].`

B.3.4.14. null_abmap()

NAME `null_abmap() -- returns an alpha-beta map object with all values set to 0 and all pointers set to NULL.`

SYNOPSIS

```
AlphaBetaMap null_contour( void );
```

INPUTS void

RETURN VALUE A new alpha-beta map object initialized to zeros and NULLs.

SEE ALSO `default_from_file() [§B.3.4.8], default_abmap() [§B.3.4.7], init_abmap() [§B.3.4.11], release_abmap() [§B.3.4.19].`

B.3.4.15. null_area_function()

NAME `null_area_function() -- returns an area function object with all values set to 0 and all pointers set to NULL.`

SYNOPSIS

```
AreaFunction null_area_function( void );
```

RETURN VALUE A new AreaFunction object with all values set to 0 and all pointers set to NULL.

SEE ALSO `init_area_function()` [§B.3.4.12], `release_area_function()` [§B.3.4.20].

B.3.4.16. null_contour()

NAME `null_contour() -- returns a contour object with all values set to 0 and all pointers set to NULL.`

SYNOPSIS

```
Contour null_contour( void );
```

RETURN VALUE A new Contour object with all values set to zeros or NULLs.

SEE ALSO `init_contour()` [§B.3.4.13], `release_contour()` [§B.3.4.21].

B.3.4.17. null_grid()

NAME `null_grid() -- returns a grid object with all values set to 0 and all pointers set to NULL.`

DESCRIPTION [TODO]

SYNOPSIS

```
SemipolGrid null_grid( void );
```

RETURN VALUE A new grid object initialized to zeros and NULLs.

SEE ALSO `init_from_file()` [§??], `default_grid()` [§B.3.4.9], `update_grid()` [§B.3.4.25], `release_grid()` [§B.3.4.22].

B.3.4.18. null_model()

NAME `null_model() -- Returns a model object with all values set to 0 and all pointers set to NULL.`

SYNOPSIS

```
Model null_model( void );
```

INPUTS void

RETURN VALUE A model object with all values set to 0 and all pointers set to NULL.

SEE ALSO `default_from_file()` [§B.3.4.8], `default_model()` [§B.3.4.10], `release_model()` [§B.3.4.23].

B.3.4.19. release_abmap()

NAME `release_abmap() -- de-allocates all memory allocated inside an alpha-beta map object and sets the corresponding pointers to NULL.`

INPUTS

- AlphaBetaMap *abMap : a pointer to the alpha-beta map object to free

RETURN VALUE 0 if everything OK, < 0 otherwise.

SEE ALSO `default_from_file()` [§B.3.4.8], `default_abmap()` [§B.3.4.7], `null_abmap()` [§B.3.4.14], `init_abmap()` [§B.3.4.11].

B.3.4.20. release_area_function()

NAME `release_area_function() -- de-allocates all memory allocated inside an AreaFunction object and sets the corresponding pointers to NULL.`

INPUTS

- AreaFunction *af : a pointer on the area function object to free

RETURN VALUE 0 if everything OK, < 0 otherwise.

SEE ALSO `init_area_function()` [§B.3.4.12], `null_area_function()` [§B.3.4.15].

B.3.4.21. release_contour()

NAME `release_contour() -- de-allocates all memory allocated inside a contour object and sets the corresponding pointers to NULL.`

INPUTS

- Contour *vtc : a pointer to the contour object to free

RETURN VALUE 0 if everything OK, < 0 otherwise.

SEE ALSO `init_contour()` [§B.3.4.13], `null_contour()` [§B.3.4.16].

B.3.4.22. release_grid()

NAME `release_grid() -- de-allocates all memory allocated inside a grid object and sets the corresponding pointers to NULL.`

SYNOPSIS

```
int release_grid( SemipolGrid *grid );
```

INPUTS

- SemipolGrid *grid : a pointer on the grid object to free

RETURN VALUE 0 if everything OK, < 0 otherwise.

SEE ALSO `default_from_file()` [§B.3.4.8], `default_grid()` [§B.3.4.9], `null_grid()` [§B.3.4.17], `update_grid()` [§B.3.4.25].

B.3.4.23. release_model()

NAME `release_model() -- de-allocates all memory allocated inside a model object and sets the corresponding pointers to NULL.`

INPUTS

- Model *model : a pointer on the model object to free

RETURN VALUE 0 if everything OK, < 0 otherwise.

SEE ALSO `default_from_file()` [§B.3.4.8], `default_model()` [§B.3.4.10], `null_model()` [§B.3.4.18].

B.3.4.24. shape_to_contour()

NAME `shape_to_contour() -- Projects the shape resulting from linear combination into the 2D plane.`

DESCRIPTION [TODO]

SYNOPSIS

```
int shape_to_contour( const Model model,
                      const SemipolGrid grid,
                      Contour *vtContour );
```

INPUTS

- const Model model : a model holding projected shapes
- const Semipolgrid grid : a semipolar grid
- Contour *vtContour : a pointer on a previously allocated vocal tract contour

OUTPUTS

- Contour *vtContour : a filled vocal tract contour.

RETURN VALUE 0 if everything OK, < 0 otherwise.

B.3.4.25. update_grid()

NAME `update_grid() -- recomputes the grid points positions after changing one or several parameters of the grid.`

DESCRIPTION [TODO]

SYNOPSIS

```
int update_grid(SemipolGrid *grid);
```

INPUTS

- SemipolGrid *grid : a pointer on the grid object to update

OUTPUTS

- SemipolGrid *grid : the updated grid object

RETURN VALUE 0 if everything OK, < 0 otherwise.

NOTES *Changing the grid length is possible (adequate memory reallocations are automatically done).*

SEE ALSO `default_from_file()` [§B.3.4.8], `default_grid()` [§B.3.4.9], `null_grid()` [§B.3.4.17], `release_grid()` [§B.3.4.22].

B.4 Module lart_def

NAME `lart_def -- Default values for initializations of structures in module libart [§B.3].`

USAGE

```
#include <lart_def.h>
```

NOTES *This header is readily included in libart.c.*

AUTHOR *Sacha Krstulović*

B.4.1 Constants

B.4.1.1. Default values for alpha-beta transform

NAME `Default values for alpha-beta transform.`

REFERENCES Those values for the alpha-beta map come from:

Heinz, J.M. and Stevens, K.N. "On the relations between lateral cineradiographs, area functions and acoustic spectra of speech" Proc. 5th International Congress of Acoustic, A44, Liege, 1965

SOURCE

```
#define DEFAULT_ABMAP_NUMCOEFFS 31

#define DEFAULT_ALPHA { 1.8, 1.8, 1.8, 1.8, \
                1.8, 1.8, 1.8, 1.8, \
                1.8, 1.8, 1.8, 1.8, \
                1.8, 1.8, 1.8, 1.8, \
                1.8, 1.7, 1.7, 1.7, \
                1.7, 1.7, 1.7, 1.7, \
                1.7, 1.7, 1.8, 1.8, \
                1.9, 2.0, 2.6 }

#define DEFAULT_BETA { 1.2, 1.2, 1.2, 1.2, \
               1.2, 1.2, 1.2, 1.2, \
               1.2, 1.2, 1.2, 1.2, \
               1.3, 1.4, 1.4, 1.5, \
               1.5, 1.5, 1.5, 1.5, \
               1.5, 1.5, 1.5, 1.5, \
               1.5, 1.5, 1.5, 1.5, \
               1.5, 1.5, 1.5 }
```

B.4.1.2. Other default constants

NAME `Other default constants.`

SOURCE

```
#define DEFAULT_GLOTTIS_AREA 1.5    /* cm^2 */
#define DEFAULT_VT_LENGTH    17.5   /* cm */
```

B.4.1.3. Default constants for semipolar grid

NAME `Default constants for semipolar grid.`

SOURCE

```
#define DEFAULT_DL 0.5
#define DEFAULT_R  5
#define DEFAULT_OMEGA_DEG -11.25
#define DEFAULT_THETA_DEG  11.25
#define DEFAULT_NUMLINES_BACK     11
/* #define DEFAULT_NUMLINES_BACK      8 */
#define DEFAULT_NUMLINES_POL      11
#define DEFAULT_NUMLINES_FRONT     6
#define DEFAULT_XORIG      0.0
#define DEFAULT_YORIG      0.0
#define DEFAULT_PHR_SCALE  1.0
#define DEFAULT_MTH_SCALE  1.0
```

B.4.1.4. Default values for model components

NAME `Default values for model components.`

DESCRIPTION See actual lart_def.h file.

NOTES *All means and variances are given in *centimeters*. The Tek-to-cm convertion has been applied to Maeda's original pb1_spec data. To go back to Maeda's original measures in pixels, multiply by TEKvt = 188.679245 .*

B.5 Application extract

NAME `extract -- Feature extraction application.`

USAGE

```
extract [-Ehcdeuvz] [-dd] [-C config_file_name]  [-L list_file_name]
     [-O output_file_name] [-R residual_file_name] [-S scale_factor]
     [-e estimator] [-f output_format] [-m model] [-f output_format]
     [-g glottis_area] [-i input_waveform_format]  [-m model]
     [-p preemp_coeff] [-r target_rate] [-s input_sample_freq]
     [-t output_coeff_type] [-w window_length] [infile]
```

DESCRIPTION Extraction of various parametric features from a signal, using various configurations of lattice filters.

Speech features output can be in HTK format.

Detail of the switches:

- -E: compute an energy coefficient and append it to each output frame.
- -C config_file_name: for experimental purposes only.
- -L list_file_name: name of a file that holds a blank-separated list of I/O files, one couple of files per line, of the form:

  inputFile1 outputFile1
  inputFile2 outputFile2
  etc.
- -O output_file_name: name of the output file that will receive the feature frames. Name '-' means stdout. [DEFAULT: no output of the features.]
- -R err_file_name: the name of the output file that will receive the output residual error, coded as double's. [DEFAULT: no error output]
- -S scale_factor: Scaling factor applied when transforming the extracted area function to a profile in the semipolar grid. [DEFAULT: 0.5]
- -c: clip the last frame if the corresponding observation window can't be entirely filled. The default is to complete the observation window with zeros and computing the corresponding feature frame.
- -d: append derivatives to the feature frame.
- -dd: append derivatives and accelerations to the feature frame.
- -e estimator: estimator for the reflection coefficients. One of burg, itakura, forward or backward.
- -f output_format:

  raw -> headerless, feature frames are output as a sequential series of float's [DEFAULT]

  htk -> features are output in the HTK format.
- -g glottis_area: a double giving the area of the glottis. [DEFAULT: 1.5cm2]
- -h: output this help and exit.
- -i input_waveform_format: 'short', 'short_swapped' or 'double' according to waveform coding of the input file. [DEFAULT: short]
- -m model: a string describing the NUT model extraction method, of the type: TYPE_Description.

  The TYPE is a letter describing what to do:
  L -> plain LPC order,
  T -> use a given NUT tube model,
  E -> exhaustive frame-based topology optimization,
  G -> global optimization on the whole sound file,
  O -> experimental, pruned optimization.

  T_ must be followed by a model description string giving the delay length of each section. For example:
  DRM would be '8/30:3,2,4,6,6,4,3,2.'
  30 even sections would be '30/30:30x1.'
  DRM could also be '8/30:3,2,4,2x6,4,3,2.'

  [DEFAULT: '30/30:30x1.']

  L_ must be given by the LPC order of a classical, eaqul-length model.

E_, G_ or O_ must be followed by a specification of the form:
number_of_unit_sections,min_length,max_length,number_of_dofs.

Examples:
- to search all the possible [8/22] configs on a frame-by-frame basis, use -m E_22,1,22,7;
- to perform plain LPC analysis of order 12, use -m L_12;
- to extract the DRM-NUT parameters, use -m T_8/30:3,2,4,6,6,4,3,2.

- -p preem_coeff: pre-emphasis coeff, a double between 0.0 and 1.0. [DEFAULT: no pre-emphasis.]
- -r target_rate: frame rate for the output, in *number of samples* with respect to the indicated input sample frequency. [DEFAULT 300, i.e. 10ms @ 30kHz]
- -s input_samp_freq: sample frequency of the input waveform, in Hz. [DEFAULT 30000Hz]
- -t output_coeff_type: one of
  lprefc: reflection coefficients [DEFAULT]
  lprefc: reflection coefficients with no constant zeros
  ar: area ratios
  lar: log area ratios
  area: area function
  shape: vocal tract profile shape before projection on a semi-polar grid [NOT FULLY FUNCTIONAL]
  diam: vocal tract diameters along a semi-polar grid [NOT FULLY FUNCTIONAL]
  prof: profile shape outline after projection in a semi-polar grid
  prof_lsq: profile shape outline after projection in a semi-polar grid and least squares smoothing in Maeda's basis
  lpc: Linear Prediction coefficients
  spec_lpc:$NSPEC : LPC spectrum, where $NSPEC gives the number of desired points per spectral frame
  spec_lpc:$NSPEC : spectrum issued by an acoustic model including losses, where $NSPEC gives the number of desired points per spectral frame
  cep:$NCEP : cepstrum, where $NCEP gives the number of desired cepstral coeffs per frame.
- -u: output usage and exit.
- -v: verbose.
- -w window_length: length of the observation window, in *number of samples* with respect to the indicated input sample frequency. [DEFAULT 750, i.e. 25ms @ 30kHz]
- -z: subtract mean from computed parameters.
- infile: name of the input file. This file must contain headerless, pcm waveform input, coded as short's or double's (to be precised with switch -i).

AUTHOR *Sacha Krstulović*

B.6 Module appgen

NAME `appgen.h -- Interface to various routines used by the application "extract".`

DESCRIPTION Most of the routines used by ”extract” are implmented in this separate module, to provide a possibility for re-use in other applications.

SEE ALSO `extract` [§B.5]

AUTHOR *Sacha Krstulović*

B.7 Module optim

NAME `optim.h -- Interface to routines for optimization of NUTs configs.`

DESCRIPTION Interface to the implementation of the exhaustive search in various forms.

SEE ALSO `extract` [§B.5]

AUTHOR *Sacha Krstulović*

Bibliography

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999). *LAPACK Users' Guide, Third Edition*. SIAM, Philadelphia.

Anderson, T. W. (1971). *The statistical analysis of time series*. Wiley, New York.

Atal, B. S. (1974). Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and recognition. *J. Acoust. Soc. Am.*, 55(6):1304–1312.

Atal, B. S., Chang, J. J., Mathews, M. V., and Tukey, J. W. (1978). Inversion of articulatory-to-acoustic transformation in the vocal tract by a computer-sorting technique. *Journal of the Acoustical Society of America*, 63(5):1535–1555.

Atal, B. S. and Rioul, O. (1989). Neural networks for estimating articulatory positions from speech. *Journal of the acoustical Society of America*, 86:123–131. suppl. 1, S67, CC1.

Badin, P., Bailly, G., Raybaudi, M., and Segebarth, C. (1998). A three-dimensional linear articulatory model based on MRI data. In *Proc. 3^{rd} ESCA/COCOSDA International Workshop on Speech Synthesis*, pages 249–254.

Badin, P. and Fant, G. (1984). Notes on vocal tract computation. Technical Report 2-3, Speech Transmission Lab., KTH.

Basseville, M. (1989). Distance measures for signal processing and pattern recognition. *Signal Processing*, (18):349–369.

Beautemps, D., Badin, P., Bailly, G., Galván, A., and Laboissière, R. (1996). Evaluation of an articulatory acoustic model based on a reference subject. In *1st ESCA Tutorial and Research Workshop on Speech Production Modeling*.

Bellanger, M., Bonnerot, G., and Coudreuse, M. (1976). Digital filtering by polyphase network : application to sample rate alteration and filter banks. *IEEE transactions on Acoustics, Speech and Signal Processing*, ASSP-24(2).

Boë, L., Perrier, P., Guérin, B., and Schwartz, J. (1989). Maximal vowel space. In *Proc. Eurospeech 89*, volume II, pages 281–284.

Bonder, L. J. (1983). The n-tube formula and some of its consequences. *Acustica*, 52:216–226.

Bothorel, A., Simon, P., Wioland, F., and Zerling, J. P. (1986). Cinéradiographie des voyelles du français. Technical report, Institut de Phonétique de Strasbourg.

Bourlard, H., Hermansky, H., and Morgan, N. (1996). Towards increasing speech recognition error rates. *Speech Communication*, 18(3):205–231.

Browman, C. P. and Goldstein, L. (1987). Towards an articulatory phonology. *Phonology yearbook*, (3):219–252.

Browman, C. P. and Goldstein, L. (1990). Gestural specification using dynamically defined articulatory structures. *Journal of Phonetics*, 18:299–320.

Burg, J. P. (1978). A new analysis technique for time series data. In Childers, D. G., editor, *Modern Spectrum Analysis.* IEEE Press/John Wiley and Sons.

Carré, R., Descout, R., and Wajskop, M., editors (1977). *Articulatory modeling and phonetics.* GALF Speech Communication Group. Proceedings of the symposium held in Grenoble, France, July 1-12 1977.

Cartsens, B. and Carstens, B. (2001). Carstens Medizinelektronik, manufacturers of the ema. Web site : `http://www.articulograph.de/`.

Chan, C. and Leung, S. (1991). A vocoder using high-order lpc filters with very few non-zero coefficients. In *Eurospeech'91*, volume 2, pages 839–842, Genova, Italy.

Charpentier, F. (1984). Determination of the vocal tract shape from the formants by analysis of the articulatory-to-acoustics nonlinearities. *Speech Communication*, 3:291–308.

Ciocea, S. and Schoentgen, J. (1998). Semi-analytic formant-to-area mapping. *Études et Travaux*, (2).

Cole, R., Noel, M., Lander, T., and Durham, T. (1995). New telephone speech corpora at CSLU. In *Proceedings Eurospeech'95*, volume 1, pages 821–824.

Deng, L. and Sun, D. X. (1994). A statistical approach to automatic speech recognition using the atomic speech units constructed from overlapping articulatory features. *J. Acoust. Soc. Am.*, 95(5):2702–2719.

Dupont, S. and Luettin, J. (2000). Audio-visual speech modelling for continuous speech recognition. *IEEE Transactions on Multimedia.* to appear.

Elliott, D. F., editor (1987). *Handbook of Digital Signal Processing.* Academic Press.

Erler, K. and Deng, L. (1992). HMM representation of quantized articulatory features for recognition of highly confusible words. In *ICASSP '92*, volume I, pages 545–548.

Erler, K. and Freeman, G. H. (1996). An hmm-based speech recognizer using overlapping articulatory features. *J. Acoust. Soc. Am.*, 100(4):2500–13.

Fant, G. (1973). *Speech Sounds and Features.* MIT Press, Cambridge.

Flanagan, J. L. (1972). *Speech analysis, synthesis and perception.* Springer Verlag.

Flanagan, J. L., Ishizaka, K., and Shipley, K. L. (1980). Signal models for low bit-rate coding of speech. *Journal of the acoustical Society of America*, 68:780–791.

Frank, W. and Lacroix, A. (1986). Improved vocal tract models for speech synthesis. In *ICASSP'86*, volume 3, pages 2011–2014, Tokyo, Japan.

Frankel, J. and King, S. (2001). Speech recognition in the articulatory domain : investigating an alternative to acoustic hmms. In *Proc. Workshop for Innovation in Speech Processing, WISP'2001.*

Friedlander, B. (1982). Lattice filters for adaptive processing. *Proc. of the IEEE*, 70(8):829–867.

Ghahramani, Z. (1993). Solving inverse problems using an em approach to density estimation. In Moser, M., Smolensky, P., Touretzky, D., Elman, J., and Weigend, A., editors, *Proc. 1993 Connectionist Models Summer School*, pages 316–323. Erlbaum Associates.

Golub, G. H. and Van Loan, C. F. (1983). *Matrix computations.* Johns Hopkins Univ. Press, 1996 (3rd) edition.

Gray, A. H. and Markel, J. D. (1976). Distance measures for speech processing. *IEEE trans. on Acoustics, Speech and Signal Processing*, ASSP-24(5):380–391.

Haykin, S. (1996). *Adaptive Filter Theory.* Prentice-Hall.

Heinz, J. M. and Stevens, K. N. (1965). On the relations between lateral cineradiographs, area functions, and acoustic spectra of speech. In *Proc. 5th Int. Congress of Acoustics*, volume A44.

Hermansky, H. (1990). Perceprual linear predictive (plp) analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738–52.

Hermansky, H. and Morgan, N. (1994). Rasta processing of speech. *IEEE Transactions on Speech and Acoustics*, 2(4):587–589.

Hieronymus, J. L. (1993). ASCII phonetic symbols for the world's languages : Worldbet. Technical report, AT&T Bell Labs, Murray Hill, NJ 07974, USA. Available in electronic form as `ftp://speech.cse.ogi.edu/pub/docs/worldbet.ps`.

Hogden, J. (1996). Improving on hidden markov models: an articulatorily constrained, maximum likelihood approach to speech recognition and speech coding. Technical Report LA-UR-96-3945, Los Alamos National Laboratory.

Itakura, F. and Saito, S. (1972). On the optimum quantization of feature parameters in the parcor speech synthesizer. In *IEEE Conference records, 1972 conference on Speech Communication*.

Jospa, P., Socquet, A., and Saerens, M. (1994). *Levels in speech communication relations and interactions*, chapter Variational formulation of the acoustico-articulatory link and the inverse mapping by means of a neural network, pages 103–113. Elsevier, Amsterdam.

Juang, B. H. and Rabiner, L. R. (1985). Mixture autoregressive hidden Markov models for speech signals. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(6):1404–1413.

King, S. and Wrench, A. (1999). Dynamical system modelling of articulator movement. In *Proc. ICPhS'99*.

Krane, M. H., Sinder, D. J., and Flanagan, J. L. (2000). Aeroacoustic modeling of speech sound production. In *Proc.* 5^{th} *Seminar on Speech Production (SPS5)*, pages 177–180, Kloster Seeon, Bavaria.

Krstulović, S. (1996). Présentation du modèle DRM. IDIAP-Com 96-3, IDIAP.

Krstulović, S. (1997). Investigation of a possible process identity between DRM and linear filtering. IDIAP-RR 97-19, IDIAP.

Krstulović, S. (1998). Acoustico-articulatory inversion of unequal-length tube models through lattice inverse filtering. IDIAP-RR 98-16, IDIAP.

Krstulović, S. (1999). LPC-based inversion of the DRM articulatory model. In *Proc. Eurospeech'99*, volume 1, pages 125–128, Budapest, Hungary.

Krstulović, S. (2000a). LPC modeling with speech production constraints. In *Proc. 5th Speech Production Seminar (SPS5)*, Kloster Seeon, Bavaria.

Krstulović, S. (2000b). Relating LPC modeling to a factor-based articulatory model. In *Proc. ICSLP 2000*, volume III, pages 67–70, Beijing, China.

Krstulović, S. (2001a). Session 1/2: Introduction to Gaussian statistics and pattern recognition. Available from the web as `ftp://ftp.idiap.ch/pub/sacha/labs/Session1.tgz`. Lab sessions given in relation to Hervé Bourlard's Speech Recognition course at EPFL (*École Polytechnique Fédérale de Lausanne*), second semester 2001.

Krstulović, S. (2001b). Session 2/2: Introduction to hidden markov models. Available from the web as `ftp://ftp.idiap.ch/pub/sacha/labs/Session2.tgz`. Lab sessions given in relation to Hervé Bourlard's Speech Recognition course at EPFL (*École Polytechnique Fédérale de Lausanne*), second semester 2001.

Krstulović, S. and Bimbot, F. (2000). Inverse lattice filtering of speech with adapted non-uniform delays. In *Proc. ICSLP 2000*, volume IV, pages 660–663, Beijing, China.

Krstulović, S. and Bimbot, F. (2001). Signal modeling with non-uniform topology lattice filters. In *Proceedings ICASSP'2001*, volume II, pages 845–848, Salt Lake City, USA.

Kunt, M. (1996). *Traitement numérique des signaux*, volume XX of *Traité d'Électricité*. Presses Polytechniques et Universitaires Romandes.

Laboissière, R. (1992). *Préliminaires pour une robotique de la communication parlée: inversion et contrôle d'un modèle articulatoire de conduit vocal.* PhD thesis, Institut de la Communication Parlée, Grenoble.

Laboissière, R., Schwartz, J.-L., and Bailly, G. (1991). Motor control for speech skills: a connectionist approach. In Touretzky, Elman, Sejnowski, and Hinton, editors, *Proceedings of the 1990 Connectionnist Models Summer School*, pages 319–327.

Lacroix, A. and Makai, B. (1979). A novel vocoder concept based on discrete time acoustic tubes. In *ICASSP'79*, pages 73–76, Washington DC.

Ladefoged, P. and Maddieson, I. (1996). *The sounds of the world's languages.* Oxford University Press / Blackwell.

Lander, T. (1997). The CSLU labeling guide. Technical report, Center for Spoken Language Understanding, Oregon Graduate Institute. Available in electronic form as `ftp://speech.cse.ogi.edu/pub/ftp_from_cse/pub/docs/labeling_conventions.ps`.

Laprie, Y. and Mathieu, B. (1998). A variational approach for estimating vocal tract shapes from the speech signal. In *Proc. ICASSP'98*, pages 929–932.

Lecuit, V. and Socquet, A. (1996). Conséquences acoustiques du passage de la coupe sagittale a la fonction d'aire. In *Proc. XXIe JEP.*

Levinson, S. E. and Schmidt, C. E. (1983). Adaptive computation of articulatory parameters from the speech signal. *Journal of the acoustical Society of America*, 74:1145–1154.

Liberman, A. M., Cooper, F. S., Shankweiler, D., and Studdert-Kennedy, M. (1967). Perception of the speech code. *Psychological Review*, (74):431–461.

Liberman, A. M. and Mattingly, I. G. (1985). The motor theory of speech perception revised. *Cognition*, (21):1–36.

Linguistic Data Consortium (1997). Catalog entry for the Switchboard-1 release 2 database. Web page: `http://www.ldc.upenn.edu/Catalog/LDC97S62.html`.

Lockwood, P. and Boudy, J. (1991). Experiments with a non-linear spectral subtractor (NSS), hidden markov models and the projection, for robust speech recognition in cars. In *Proc. Eurospeech '91*, pages 79–82.

Luettin, J. (1999). Speech reading. In Noyes, J. and Cooke, M., editors, *Modern Interface Technology: The Leading Edge*, pages 97–121. Research Studies Press Ltd.

Maddieson, I. (1984). *Patterns of sounds*. Cambridge University Press.

Maeda, S. (1979). Un modèle articulatoire de la langue avec des composantes linéaires. In *Actes des 10èmes journées d'études sur la parole*, pages 154–162.

Maeda, S. (1990). *Speech production and speech modelling*, chapter Compensatory articulation during speech : evidence from the analysis and synthesis of vocal tract shapes using an articulatory model, pages 131–149. NATO ASI series. Kluwer Academic.

Makhoul, J. (1975a). Linear Prediction: a tutorial review. *Proceedings of the IEEE*, 63(4):561–580.

Makhoul, J. (1975b). Spectral Linear Prediction : properties and applications. *IEEE trans. on Acoustics, Speech and Signal Processing*, ASSP-23(3):283–296.

Makhoul, J. (1977). Stable and efficient lattice methods for linear prediction. *IEEE trans. on Acoustics, Speech and Signal Processing*, ASSP-25(5):423–428.

Markel, J. D. and Gray, A. H. (1976). *Linear prediction of speech*. Springer-Verlag.

Matsuzaki, H. and Motoki, K. (2000). Fem analysis of 3-d vocal tract model with asymmetrical shape. In *Proc.* 5^{th} *Seminar on Speech Production (SPS5)*, pages 329–332, Kloster Seeon, Bavaria.

Mermelstein, P. (1967). Determination of the vocal-tract shape from measured formant frequencies. *Journal of the acoustical Society of America*, 41(5):1283–1294.

Mokbel, C., Mauuary, L., Karray, L., Jouvet, D., Monné, J., Simonin, J., and Bartkova, K. (1997). Towards improving ASR robustness for PSN and GSM telephone applications. *Speech Communication*, 23(1-2):141–159.

Mokhtari, P. (1998). *An acoustic-phonetic articulatory study of speech-speaker dichotomy*. PhD thesis, Univ. of South Wales, Canberra, Australia.

Mokhtari, P. and Clermont, F. (2000). New perspectives on linear-prediction modelling of the vocal tract : uniqueness, formant-dependence and shape parameterisation. In *Proc.* 8^{th} *Australian International Conference on Speech Science and Technology (SST-2000)*, Canberra, Australia.

Morse, P. M. and Ingard, K. U. (1968). *Theoretical acoustics*. Mc Graw-Hill.

Motoki, K., Badin, P., Pelorson, X., and Matsuzaki, H. (2000). A modal parametric method for computing acoustic characteristics of three-dimensional vocal tract models. In *Proc.* 5^{th} *Seminar on Speech Production (SPS5)*, pages 325–328, Kloster Seeon, Bavaria.

Mrayati, M. (1976). *Contribution aux études sur la production de la parole : modèles éléctriques du conduit vocal avec pertes, du conduit nasal et de la source vocale; études de leurs interactions; relations entre disposition articulatoire et caractéristiques acoustiques.* PhD thesis, I.N.P. Grenoble.

Mrayati, M., Carré, R., and Guérin, B. (1988). Distinctive regions and modes: a new theory of speech production. *Speech Communication*, (7):257–286.

Munhall, K. G., Vatikiotis-Bateson, E., and Tokhura, Y. (1995). X-ray film database for speech research. *Journal of the Acoustical Society of America*, 98(2):1222–1224.

Oppenheim, A. V. and Schafer, R. W. (1989). *Discrete-time signal processing.* Prentice Hall.

Ouni, S. and Laprie, Y. (2000). Improving acoustic-to-articulatory inversion by using hypercube codebooks. In *Proc. ICSLP 2000*, Beijing, China.

Painter, T. and Spanias, A. (2000). Perceptual coding of digital audio. *Proceedings of the IEEE*, 88(4):451–513.

Papcun, G., Hochberg, J., Thomas, T. R., Laroche, F., Zacks, J., and Levy, S. (1992). Inferring articulation and recognizing gestures from acoustics with a neural network trained on x-ray microbeam data. *Journal of the acoustical Society of America*, 92:688–700.

Perrier, P., Boë, L. J., and Sock, R. (1992). Vocal tract area function estimation from midsagittal dimensions with CT scans and a vocal tract cast : modelling the transition with two sets of coefficients. *Journal of Speech and Hearing Research*, (35):53–67.

Pitrelli, J., Fong, C., Wong, S. H., Spitz, J. R., and Lueng, H. (1995). Phonebook : a phonetically rich isolated-word telephone speech database. In *Proc. ICASSP'95*, pages 101–104.

Poritz, A. B. (1982). Linear predictive hidden Markov models and the speech signal. In *Proc. ICASSP'82*, pages 1291–1294, Paris, France.

Prado, P. P. L., Shiva, E. H., and Childers, D. G. (1992). Optimization of acoustic-to-articulatory mapping. In *ICASSP '92*, volume 2, pages 33–36.

Quackenbush, S. R., Barnwell, T. P., and Clements, M. A. (1988). *Objective Measures of Speech Quality.* Prentice Hall, Englewood Cliffs, NJ.

Rabiner, L. and Juang, B. H. (1993). *Fundamentals of Speech Recognition.* Prentice Hall.

Rahim, M. G. and Goodyear, C. C. (1990). Estimation of vocal-tract filter parameters using a neural net. *Speech Communication*, 9:49–55.

Rahim, M. G., Goodyear, C. C., Kleijn, W. B., Schroeter, J., and Sondhi, M. M. (1993). On the use of neural networks in articulatory speech synthesis. *Journal of the acoustical Society of America*, 93(2):1109–1121.

Ramsay, G. (1996). A non-linear filtering approach to stochastic training of the articulatory-acoustic mapping using the em algorithm. In *ICSLP '96*.

Richardson, M., Bilmes, J., and Diorio, C. (2000a). Hidden-articulator markov models for speech recognition. In *Proc. ASR'2000 Workshop*.

Richardson, M., Bilmes, J., and Diorio, C. (2000b). Hidden-articulator markov models : performance improvements and robustness to noise. In *Proc. ICSLP'2000*, Beijing, China.

Richmond, K. (2001). Mixture density networks, human articulatory data and acoustic-to-articulatory inversion of continuous speech. In *Proc. Workshop on Innovation in Speech Processing, WISP'2001*.

Robinson, E. A. and Durrani, T. S. (1986). *Geophysical signal processing*. Prentice Hall.

Rose, R. C., Schroeter, J., and Sondhi, M. M. (1994). An investigation of the potential role of speech production models in automatic speech recognition. In *ICSLP '94*, volume S12-1, pages 575–578.

Rosenberg, A. (1971). Effects of the glottal pulse shape on the quality of natural vowels. *Journal of the Acoustical Society of America*, 49(2):583–590.

Scaife, R. (1989). Vocal tract area estimation - extending the Wakita inverse filter. In *Eurospeech '89*, pages 648–651.

Schmidbauer, O. (1989). Robust statistic modelling of systematic variabilities in continuous speech incorporating acoustic-articulatory relations. In *ICASSP '89*, volume 1, pages 616–619.

Schroeder, M. R. (1967). Determination of the geometry of the human vocal tract by acoustic measurements. *Journal of the Acoustical Society of America*, 41(4):1002–1010.

Schroeder, M. R. and Atal, B. S. (1985). Code-excited linear predictive (CELP) : High quality speech at very low bit rates. In *Proc. ICASSP'85*, pages 937–940.

Schroeter, J., Meyer, P., and Parthasarathy, S. (1990). Evaluation of improved articulatory codebooks and codebook access methods. In *ICASSP '90*, volume 1, pages 393–396.

Schroeter, J. and Sondhi, M. M. (1989). Dynamic programming search of articulatory codebooks. In *Proceedings of ICASSP'89*, volume 1, pages 588–591.

Schroeter, J. and Sondhi, M. M. (1994). Techniques for estimating vocal-tract shapes from the speech signal. *IEEE Transactions on Speech and Audio Processing*, 2(1, part II):133–150.

Shirai, K. and Honda, M. (1980). Estimation of articulatory motion from speech waves and its application for automatic recognition. In Simon, J. C., editor, *Spoken language generation and understanding*, pages 87–99. D. Reidel Publishing Company.

Shirai, K. and Kobayashi, T. (1991). Estimation of articulatory motion using neural networks. *Journal of Phonetics*, 19:379–385.

Sondhi, M. M. (1979). Estimation of vocal tract areas: the need for acoustical measurements. *IEEE Trans. Audio, Speech and Signal Processing*, 27(3):268–273.

Soquet, A., Saerens, M., and Jospa, P. (1990). Acoustic-articulatory inversion based on a neural controller of a vocal tract model. In *Proceedings of the ESCA workshop on Speech Synthesis*, pages 71–74.

Stephenson, T., Bourlard, H., Bengio, S., and Morris, A. C. (2000). Automatic speech recognition using dynamic bayesian networks with both acoustic and articulatory variables. In *Proc. ICSLP'2000*, Beijing, China.

Story, B., Titze, I., and Hoffman, E. (1996). Vocal tract area functions from magnetic resonance imaging. *Journal of the Acoustical Society of America*, 100(1):537–554.

Sundberg, J. (1969). On the problem of obtaining area function from lateral x-ray pictures of the vocal tract. *STL-QPSR*, (1):43–45.

Välimäki, V. (1995). *Discrete-Time Modeling of Acoustic Tubes Using Fractional Delay Filters*. PhD thesis, Helsinki University of Technology, Helsinki, Finland. See `http://www.acoustics.hut.fi/~vpv/publications/vesa_phd.html`.

Viswanathan, R. and Makhoul, J. (1975). Quantization properties of transmission parameters in linear predictive systems. *IEEE trans. on Acoustics, Speech and Signal Processing*, ASSP-23:309–321.

Wakita, H. (1972). Estimation of vocal tract shape by optimal inverse filtering and acoustic/articulatory conversion methods. Technical Report 9, Speech Communication Research Lab, Santa Barbara, Cal.

Wakita, H. (1973). Direct estimation of the vocal-tract shape by inverse filtering of acoustic speech waveforms. *IEEE Transactions on Audio and Electroacoustics*, pages 417–427.

Wakita, H. (1979). Estimation of vocal-tract shapes from acoustical analysis of the speech wave: the state of the art. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 27(3):281–285.

Wakita, H. and Gray Jr., A. H. (1975). Numerical determination of the lip impedance and vocal tract area functions. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(6):574–580.

Westbury, J. H. (1994). *X-ray microbeam speech production database user's handbook*. Waisman Center, University of Wisconsin, 1.0 edition.

Wrench, A. and Richmond, K. (2000). Continuous speech recognition using articulatory data. In *Proc. ICSLP'2000*, Beijing, China.

Wrench, A. A. and Hardcastle, W. J. (2000). A multichannel articulatory speech database and its application for automatic speech recognition. In *Proc. 5th Seminar on Speech Production (SPS5) : models and data*, pages 305–308, Kloster Seeon, Germany.

Yehia, H., Rubin, P., and Vatikiotis-Bateson, E. (1997). Quantitative association of orofacial and vocal-tract shapes. In *Proc. Workshop on Audio-Visual Speech Processing*, pages 41–44, Rhodes, Greece.

Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., and Woodland, P. (1995-1999). *The HTK Book (for HTK Version 2.2)*. Entropic Ltd.

Yu, J. (1993). A method to determine the area function of speech based on perturbation theory. *STL-QPSR*, 4:77–95.

Zacks, J. and Thomas, T. R. (1994). A new neural network for articulatory speech recognition and its application to vowel identification. *Computer Speech and Language*, 8:189–209.

Zhan, P. and Waibel, A. (1997). Vocal tract length normalization for large vocabulary continuous speech recognition. Technical Report CMU-CS-97-148, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, PA 15213.

Zlokarnik, I. (1993). Experiments with an articulatory speech recognizer. In *Eurospeech '93*, pages 2215–2218.

Sacha KRSTULOVIĆ
sacha@idiap.ch
http://www.idiap.ch/~sacha

Studies

- **96-2001** : PhD student at **IDIAP**[1] (see "Professional experiences").
- **June 96** : engineering diploma from the **ESTACA**[2], **Aeronautics** branch, with a specialization in **Signal Processing**.
- **July-august 95** : summer program about Information Transfer Technologies at **Harvard, USA**. Project: implementation of a formant synthesizer under MATLAB/Simulink.
- **94-95** : engineering projects for the ESTACA:
  - application of **Neural Networks** to the classification of AR processes;
  - programmation of a **Finite Elements Method** application under MATLAB.
- **1990** : 'Baccalaureat' (french high school diploma), specialty: science ('C' series).

Professional experiences

- **Since march 97** : research assistant/PhD student at **IDIAP**, affected to the ARTIST project concerning the **modeling of the speech signal** under production constraints.
- **July-august 94** : traineeship in **Japan**, at **RICOH Inc.**, as a technical assistant to the International Marketing Group.
- **July-august 93** : traineeship in **Ukraine**, at **ECSAT Inc.**, as an assistant programmer.
- **July-august 92** : summer job at the **CCIP** (*Chambre de Commerce et d'Industrie de Paris*, French Chamber of Industry).

Miscellaneous skills

- **Languages** : **French** native speaker; **English** fluently spoken, read and written; notions of **Italian**.
- **Computer science** :
  - programming : **C, Perl, C++, FORTRAN**.
  - Excellent knowledge of **Matlab** and its toolboxes.

[1] Institut Dalle Molle d'Intelligence Artificielle Perceptive; C.P. 592; CH-1920 Martigny - Switzerland.
[2] École Supérieure des Techniques Aéronautiques et de Construction Automobile; Levallois-Perret - France.

- **Computer science** (continued) :
    - **UNIX, GNU/Linux**, LaTeX, automake, CVS etc.
    - **Windows**, MSOffice etc.

Extra-curricular activities

- **92-94** : creation and management of the **ESTACA's library**.
- **Aikido**, Black Belt 1^{st} Dan; **Capoeira** Abadá style, 3^{rd} grade (yellow).

Personal information

- Born april 13th, 1972, in Lons Le Saunier, France; French citizenship; single.
- French Military Service fulfilled.