EVALUATING THE INTELLIGIBILITY OF SPEECH CHIMAERAS BASED ON ENVELOPE AND TEMPORAL FINE STRUCTURE CUES

EVALUATING THE INTELLIGIBILITY OF SPEECH CHIMAERAS BASED ON ENVELOPE AND TEMPORAL FINE STRUCTURE CUES

BY

YUJIE LI, B.Eng.

A Thesis

submitted to the Electrical & Computer Engineering

and the School of Graduate Studies

of McMaster University

in partial fulfilment of the requirements

for the degree of

Masters of Applied Science

Masters of Applied Science (2024) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	EVALUATING THE INTELLIGIBILITY OF SPEECH CHIMAERAS BASED ON ENVELOPE AND TEMPORAL FINE STRUCTURE CUES
AUTHOR:	Yujie Li B.Eng. (Underwater Acoustic Engineering), Northwestern Polytechnical University, Xi'an, CHina
SUPERVISOR:	Dr. Ian C. Bruce
NUMBER OF PAGES:	xix, 177

Lay Abstract

In daily life, being able to hear and understand the speech of people around us is the basis for normal communication and life. In order to investigate how people use different information in the speech signal to understand speech, this thesis decomposes the speech signal into envelope and temporal fine structure, where the envelope is the contour of the signal amplitude over time, which provides information such as syllable rhythms and intonation patterns, while the temporal fine structure is the fast oscillating part of the signal, which carries pitch and timbre information. In this thesis, the envelope and temporal fine structure of different signals are combined to produce speech chimaeras as test datas, and speech intelligibility is evaluated using different algorithms to select the best performing model, in an effort to contribute to the study of speech intelligibility.

Abstract

Speech intelligibility is a measure of the human ability to understand speech signals. While the speech signal can be decomposed into its envelope and temporal fine structure, where the envelope is the contour of the signal amplitude over time, revealing the rhythm and intensity of the speech signal, the temporal fine structure is the rapidly oscillating portion of the signal that carries pitch and timbre information. Studies on speech intelligibility have shown that the acoustic envelope and the temporal fine structure contribute to speech intelligibility and play an important role in quiet and background noise, respectively.

In this thesis, two speech signals are selected, one signal retains the envelope, the other signal retains the temporal fine structure, and then the envelope of one signal is combined with the temporal fine structure of the other signal to generate different speech chimera signals.

Three methods are applied to evaluate the speech intelligibility, namely Spectro-Temporal Modulation Index (STMI), Neurogram Similarity Index Measure (NSIM), and Cross-Correlation Coefficients (CCC). This thesis describes these three methods in detail, in particular the creation of physiologically based assessment matrices, and then analyzes and compares the results by creating regression models of the predicted values of the different algorithms with experimentally measured subjective

perceptions.

This thesis shows that the combination of the STMI with either the fine-timing NSIM or the temporal fine structure CCC provides the optimal prediction model for speech chimera signals, and provides some implications for speech intelligibility research.

Key words: speech intelligibility, envelope, temporal fine structure, mean-rate, fine-timing, speech chimaeras

To the people who love me and always support me.

Acknowledgements

I would like to thank my supervisor Dr. Ian C. Bruce first, for his constant academic help in guiding me through my thesis. His wealth of knowledge and positive attitude always kept me organized and on track, especially when I am stuck. And I am also grateful to Sean Clarke for sharing his preliminary cross-correlation code. This thesis would not have been finished without their help.

I would also like to thank my family for always being with me and supporting my education wholeheartedly all the time.

Contents

List of Figures

List of Tables

Abbreviations

Abbreviations

AI	Articulation Index
AE	Articulation Error
SNR	Signal to Noise Ratio
SII	Speech Intelligibility Index
SSIM	Structural Similarity Index
AN	Auditory Nerve
ENV	Envelope
TFS	Temporal Fine-Structure
SAC	Shuffled Auto-Correlogram
SCC	Shuffled Cross-Correlogram
WGN	White Gaussian Noise

MN	Matched Noise
PSTH	Post-Stimulus Time Histogram
STMI	Spectro-Temporal Modulation Index
NSIM	Neurogram Similarity Index Measure
CCC	Cross-Correlation Coefficients
CNC	consonant-nucleus-consonant
CVC	consonant-vowel-consonant
CF	characteristic frequency

Declaration of Academic Achievement

The author wrote this thesis and conducted all of the research contained therein, under the supervision of Dr. Ian C. Bruce.

Chapter 1

Introduction & Literature Review

1.1 Introduction

Speech is the most convenient, versatile, and important carrier of information in human society. Although the physical body can also convey information, speech is still the most common means of public communication. This is why it is so important to hear and understand what is being said in social interactions. Many factors can affect our communication, such as distance, voice volume, noise level, accent, etc.
Speech intelligibility is known as a measure of how much speech information the listener perceives. It is important to note that speech quality and intelligibility are not synonymous, good intelligibility can be achieved with degraded speech quality in some cases. If speech quality is about the "how", intelligibility is about the "what". In other words, speech quality is concerned with the "how" the speech sounds - whether it is clear, lossless, noiseless, etc. While speech intelligibility is concerned with the comprehensibility of speech information, that is, whether the listener can accurately

understand the "what" in the speech signal, i.e., the conveyed messages or vocabulary. After understanding the concept of speech intelligibility, this thesis investigates various methods for predicting speech intelligibility.

The first chapter describes the importance of speech in life, lists the various factors that may affect speech intelligibility, and leads to the criteria for predicting speech intelligibility. Then the chapter reviews many methods about predicting speech intelligibility.

Chapter 2 reviews the literature of Wirtzfeld et al. (2017) and Smith et al. (2002), and presents the process of constructing Speech Chimaeras. And the design of the experiment is also clarified, regarding the setting of experimental parameters, the design of the process and the pre-determination of the results.

Chapter 3 reviews the literature of Elhilali et al. (2003), Hines and Harte (2012), Swaminathan and Heinz (2012), and so on. Then it describes in detail three methods of predicting the speech intelligibility, Spectro-Temporal Modulation Index, Neurogram Similarity Index Measure, and Cross -Correlation Coefficients. And regression models are then constructed to compare and evaluate among different algorithms.

Chapter 4 summarizes the conclusions of this thesis and suggests the future improvements of the work.

MATLAB code produced in this thesis and additional figures are presented in the Appendices.

1.2 Literature Review

Speech intelligibility is a measure of the ability to understand the spoken word under given conditions. In general, speech intelligibility can be quantified by counting the

number of correctly recognized words or phonemes. And most speech intelligibility depends on the audibility of the signal in each frequency band.

On this basis, speech intelligibility prediction initially used the Articulation Index (AI) (French and Steinberg, 1947), in which the Articulation Error (AE) in each frequency band does not affect the other bands, so each band is independent. According to this theory, the intelligibility index can then be expressed as a weighted sum of the intelligibility corresponding to each frequency component of speech. And the goal of this method is to convert the prediction of articulation into the measurement of signal to noise ratio (SNR), which refers to the difference between the normal sound signal and the noise signal (power) when there is no signal. The SNR is calculated for each frequency band, where Band-important-function (BIF) is a weighting factor for the band obtained through extensive experiments. Speech Intelligibility Index (SII) is developed as an extension of the AI. Compared to AI, SII mainly considers the masking effect which is that one of the sounds (masking sound) makes the other sound (masked sound) harder to hear when calculating SNR speech in different frequency bands (Taghavi, S. M. R., Mohammadkhani, G., & Jalilvand, H., 2022).

The Spectro-Temporal Modulation Index (STMI) is one predictor of speech intelligibility based on a physiological model. It quantifies the differences in spectral temporal modulations between a clean speech signal and a noisy or distorted speech signal using a physiologically-based cortical model (Elhilali et al., 2003, Wirtzfeld et al.,2017). In addition, there are likely more centrally located auditory processes, such as lateral inhibition networks (LINs), that may convert spike-timing cues into mean-rate cues (Shamma and Lorenzi 2013).

An alternative predictor is the the Neurogram Similarity Index Measure (NSIM). This

predictor is based on the structural similarity index (SSIM) (Wang et al., 2004), which is an index for evaluating the similarity between two images and evaluates the image quality by comparing the brightness, contrast, and structural information of images. The main innovation of NSIM is the application of SSIM to neurograms generated by auditory nerve (AN) models. The NSIM is calculated for the points over the patch or windowed area, so the overall NSIM similarity index corresponding to the two neurograms is calculated as the average of the NSIM index values over all patches. Similar to SSIM, NSIM analyzes three factors of brightness, contrast and structure in each individual pixel and compares the overall range of image pixel intensities (Hines and Harte, 2012, Wirtzfeld et al.,2017). After weighted adjustments, comparisons are made with the reference image to obtain the differences in intensity (luminance), variance (contrast), and cross-correlation (structure). The core idea of NSIM is to evaluate speech intelligibility by comparing the neurograms of the reference speech signal and the degraded speech signal.

The Cross-Correlation Coefficients is another predictor of speech intelligibility based on a physiological model. The basic idea of this method is to use auditory nerve models to simulate the response of human hearing to speech signals, generate so-called "spike trains", and then evaluate speech intelligibility by calculating the cross-correlation between these spike trains. In this methods, separate metrics for ENV and TFS are computed using shuffled auto-correlograms (SACs) and shuffled cross-correlograms (SCCs) (Joris, 2003; Louage et al., 2004; Joris et al., 2006). So the cross-correlation coefficients for both TFS and ENV are extracted at the peak.

In summary, these studies emphasize the importance of ENV and TFS for speech intelligibility, ENV is important for identifying phonemes and words in speech and

TFS is important for identifying pitch and tonal variations in speech, both are also complementary in speech signals and the combination of both helps to achieve high prediction of speech intelligibility. These profoundly reveal the contribution of time and frequency domain features to speech recognition and understanding.

Therefore, this thesis will analyze and compute the above algorithms, while comparing and discussing the combination of the algorithms in depth and comprehensively, focusing on their complementarity and enhancement, and evaluating the optimal choice for predicting speech intelligibility.

Chapter 2

Model building

2.1 ENV & TFS

The speech signal can be decomposed into envelope and temporal fine structure information. The envelope (ENV) represents the contour of the signal amplitude over time, reflecting the syllable rhythm and intonation intensity, while the temporal fine structure (TFS) refers to the rapidly oscillating part of the signal, carrying pitch and timbre information. These two components affect various aspects of auditory perception, such as loudness, pitch, timbre perception, and spatial hearing.

In general, the envelope is sufficient for conveying speech information in a quiet environment, while the temporal fine structure provides additional speech cues in a noisy environment. The envelope provides important information about the speech signal's rhythm and intensity changes in a quiet environment. Due to the low level of background noise, listeners can more easily recognize syllable boundaries, words, and sentence structures in speech, which is crucial for language comprehension. The TFS carries information about the pitch and timbre of a speech signal that, unlike

envelopes that can be masked by background noise, remains relatively clear and helps listeners distinguish between different sound sources in complex auditory scenarios, like identifying a specific speaker in a multi-person conversation.

In order to facilitate the calculation, Rosen (1992) proposed that the speech signal could be divided into three fluctuation ranges temporally. He defined the fluctuation range of about 2-50Hz as ENV, the fluctuation range of about 50-500Hz as periodicity, and the fluctuation range of about 500-10kHz as TFS. However, in many studies, the periodicity of fluctuations in the 50-500Hz range is often categorized as either ENV or TFS, or as a cutoff frequency within their range. Thus in this thesis, speech signals are also divided into ENV and TFS temporally.

Due to background noise having some effect on both the ENV and TFS cues, this thesis utilizes vocoders to manipulate the ENV and TFS of speech. Specifically, it divides the broadband speech into a set of frequency channels and decomposes the narrowband signals into the corresponding ENV and TFS components. On this basis a synthesized speech signal can be generated using certain aspects of the ENV or TFS of the original speech signal.

2.2 Speech Chimaeras

2.2.1 Chimeras generation

Speech chimaera is a synthetic speech signal generated by combining envelope and temporal fine structure of different speech signals and is designed to study how listeners use different information (e.g., intonation, rhythm, pitch, etc.) in the speech

signal to understand language (Smith et al., 2002).

The generation of the speech chimera is divided into the following steps, firstly, selecting two acoustic signals, one for ENV and the other for TFS. The two signals are then separated respectively, as described in Section 2.1, and manipulated by vocoders to generate ENV-only or TFS-only versions. Finally, these signals are reorganized in the way that the ENV of one signal is multiplied by the TFS of the other at each frequency channel and then summed across the frequency bands.

For the experiments in this thesis, one signal is the speech signal and the other is the noise. The noise signals selected are white Gaussian noise (WGN) and matched-noise (MN). Matched-noise is a special type of noise whose spectral characteristics match a given speech signal. It is generally generated based on the spectral characteristics of the speech signal, so the matched-noise of the clean speech is done by performing a Fast Fourier Transform (FFT) on each speech signal individually, preserving the magnitude spectrum, randomizing the phase (preserving the antisymmetric nature of the phase spectrum), and then taking the real part of the inverse FFT. In addition, this thesis generates pure speech TFS (Speech TFS with Flat ENV) stimuli by extracting TFS from all frequency bands. The process of generating speech chimaeras is shown in Figure 2.1 (Smith et al, 2002).

Hence, a total of five different types of speech chimeras are presented in this thesis, namely Speech ENV with MN TFS, Speech ENV with WGN TFS, Speech TFS with MN ENV, Speech TFS with Flat ENV and Speech TFS with WGN ENV.

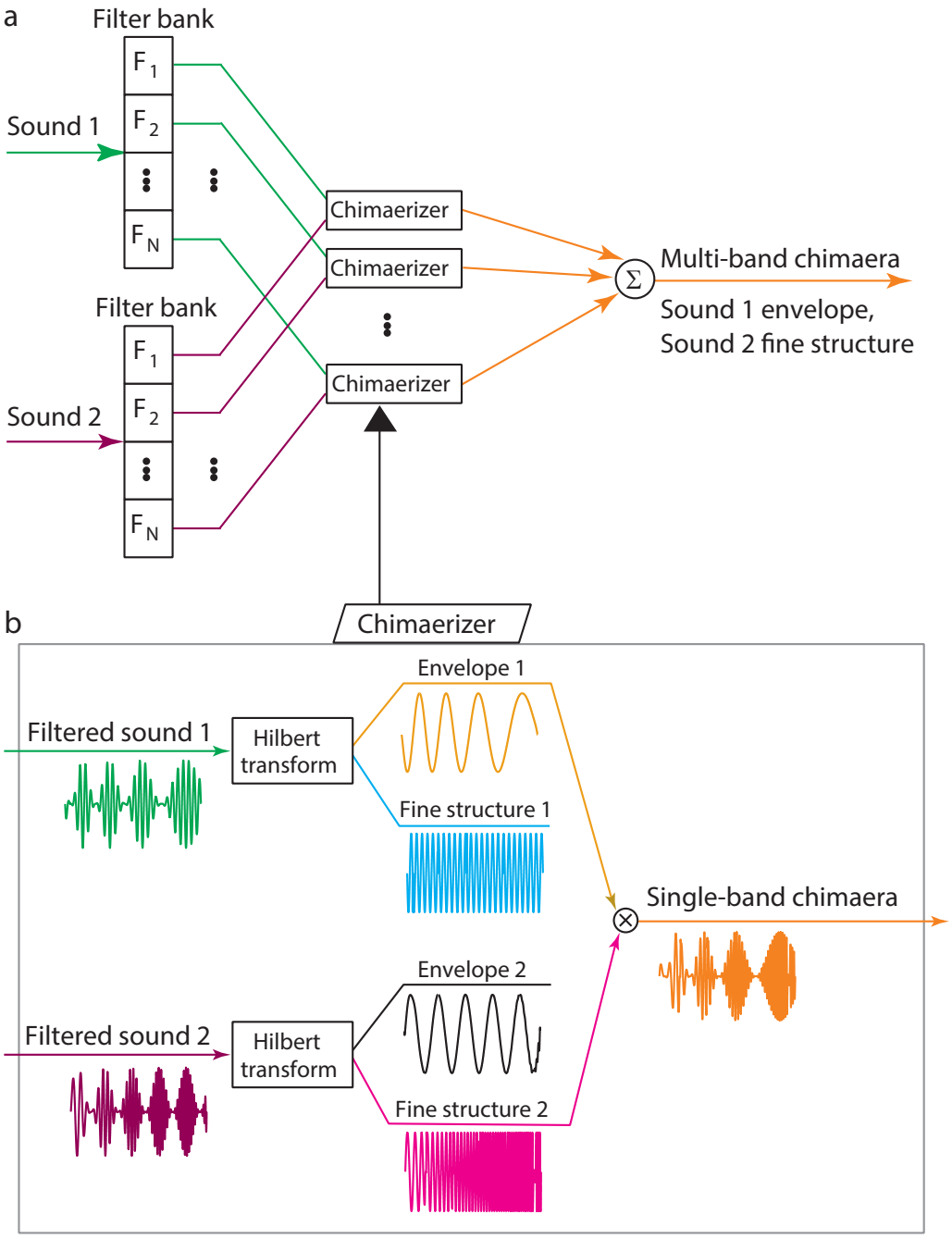

Figure 2.1: a, Two sounds are used as input. Each sound is split into N frequency bands by a filter group. The ENVs and TFSs of the two signals are swapped to produce a single-band chimaera. b, Example of a waveform in the chimaeraer, where the band-limited input signal is decomposed into an ENV and a TFS by the Hilbert transform. The product of the envelope 1 and the fine structure 2 is the single-frequency auditory chimaeras. Reprinted from Fig. 1 of from Smith et al, (2002)

2.2.2 Experiment Design

In a word recognition experiment reported in Wirtzfeld et al. (2017), five native English-speaking, normal-hearing subjects between the ages of 18 and 21 were asked to identify the final word in the sentence "Say the word (test word)". The test words were all selected from the NU-6 word list (Tillman and Carhart 1966), which contains a total of 200 monosyllabic consonant-nucleus-consonant (CNC) words, recorded by an American native English speaker (Auditec, St. Louis). While Tillman and Carhart (1966) used the term "nucleus" to describe the central phonemes because they include diphthongs as well as vowels, to simplify the description of our results we will use the term "vowel" to refer to the central phoneme. The test sentences had all undergone auditory chimera processing as described above.

The next step is the experimental procedure. The subjects were tested in a quiet room. All signals were generated with a high quality PC sound card at a sampling rate of 44,100 Hz. The sound was presented to the subjects through the amplifier and headphones. The signals were calibrated using a B & K 2260 Investigator Sound Analyzer to adjust the target speech to a presentation level of 65 dB SPL. The test was conducted without prior training and consisted of five 1-h sessions for each subject. The five different chimaera types were each tested in a different session, and the order of the chimaera types was randomized for each subject. The chimaera types were blocked in this way to allow participants to quickly become familiar with each type of processing, as the Speech ENV and Speech TFS chimaeras can sound very different (Wirtzfeld et al.,2017).

For each chimaera type, seven sets of vocoder frequency bands are used. For each set of frequency bands, 50 test words are generated. In these thesis, these 50 test words

are randomly selected from the 200 available words in the NU-6 list, resulting in 1750 test words used in that study. This set of words is presented to the subjects using the following procedure:

- Randomly select one of 350 available words (50 words for each of the 7 filter sets) for the chimaera type being tested in that session.
- Ask the subject to repeat the word as they perceived it.
- Voice records the subject's verbal response as well as a written record. Subjects are told that they might not be able to understand all the test words because the speech processing made some of them unintelligible. In the cases where a subject could not recognize a test word, they are asked to guess to the best of their ability (Wirtzfeld et al., 2017). The identification results measured from the perceptual experiment are shown in Figure 2.2.

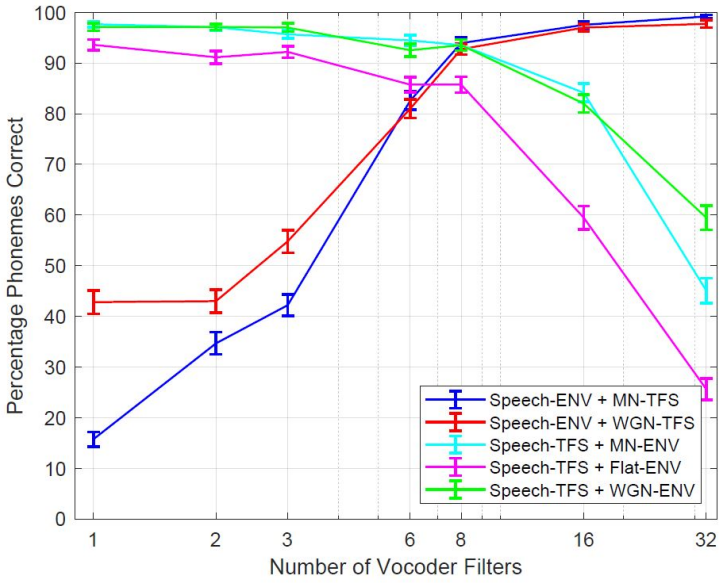

Figure 2.2: Phoneme perception scores from the listening experiment as a function of the number of vocoder filters, averaged over words and listeners. Error bars show ± 1 standard error of the mean (SEM). The Speech ENV chimaeras retains the ENV of the original speech signal and combines it with the WGN or MN TFS of another signal. Similarly, Speech TFS chimaeras retain the TFS of the original speech signal and combine it with the WGN, MN, or Flat ENV of another signal.

2.3 Neurogram Generation

For signals in speech chimaeras, the generation of the neurogram is a process of analyzing speech signals by simulating the response of the auditory periphery model. To achieve this, this thesis uses the auditory periphery model of Bruce et al. (2018) to simulate the response of auditory nerve (AN) fibers to stimuli of varying intensities. In constrast, Wirtzfeld et al. (2017) used the old auditory periphery model proposed by Zilany et al. (2009) to simulate synaptic transmission between inner ear hair cells and auditory nerve fibers. This model is capable of producing auditory nerve fiber model outputs that are consistent with observed data from mammalian hearing and is used in the study of normal and impaired hearing, as well as in the development

and evaluation of hearing aids.

The new model by Bruce et al. (2018) is a further improvement and development of it. This new model introduces some new features in the simulation of the auditory process, such as a more accurate simulation of neurotransmitter release and synaptic dynamics, which emphasizes the importance of the limited number of neurotransmitter release sites and through this feature better explains the statistical properties of spontaneous spiking of auditory nerve fibers. The model is further extended to sound-evoked nerve fiber responses and integrated into a comprehensive model of the auditory periphery.

The characteristic frequency (CF) spacing is different for the STMI and the NSIM methods. The model calculates 128 logarithmically spaced characteristic frequencies (CFs) ranging from 180 Hz to 7040 Hz for STMI, unlike the only 40 CFs from 250 Hz to 16000 Hz required by the NSIM. For each speech signal, a post-stimulus time histogram (PSTH) is calculated for each CF, counting them at 10 µs intervals. The PSTHs of the different CFs are then stacked to form a spectrogram-like representation, which is called a "neurogram".

In this thesis, the PSTH responses at each CF are generated by summing the individual PSTH responses of a set of 50 AN fibers: The set includes 30 low-threshold fibers with high spontaneous rates (>18 spikes per second), 15 medium-spontaneous-rate fibers (0.5 to 18 spikes per second), and 5 high-threshold fibers with low spontaneous rates (<0.5 spikes per second). The distributions of the fibers are consistent with previous studies (Jackson and Carney 2005; Zilany et al. 2009). This representation visually illustrates the processing of speech signals in the auditory system.

Before inputting the speech signal to the auditory model, preprocessing is necessary

to simulate typical auditory functions and meet the model's requirements. This thesis scales the stimulus signal to a 65 dB sound pressure level (SPL) presentation level, and upsamples the signal to the model's 100 kHz sampling rate.

The neurogram provides a research tool for analyzing and understanding the processing of speech signals in an auditory model. After subsequent additional processing, its alternative forms explicitly characterize the intrinsic mean rate and spike timing neural cues (Wirtzfeld et al., 2017), respectively.

Chapter 3

Methods and results

3.1 Methods

In this chapter, three methods of evaluating speech intelligibility will be discussed, which are Spectro-Temporal Modulation Index (STMI), Neurogram Similarity Index Measure (NSIM), and Cross-Correlation Coefficients (CCC).

3.1.1 STMI

STMI is a measure of how the output changes when noise, reverberation, or other distortion is applied to an acoustic signal based on the physiological model. STMI is also a cortical model that quantifies differences in spectral-temporal modulation between clean speech signals and the noisy or distorted speech signals (Chi et al., 1999; Elhilali et al., 2003). It is sensitive only to mean rate or mean neural activity. A schematic of STMI generation is shown in Figure 3.1.

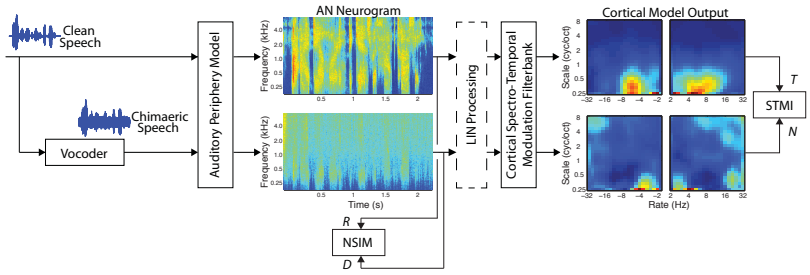

Figure 3.1: A schematic illustration of the process of STMI and NSIM based on the the AN neurograms. Reprinted from Fig. 2 of from Heinz & Swaminathan (2009).

Following the method mentioned by Wirtzfeld et al. (2017), this thesis convolves the peristimulus time histogram (PSTH) of each characteristic frequency (CF) with a rectangular window of 16 ms and processes it with 50% overlap. This produces an effective neurogram with a sampling rate of 125 Hz, which eliminates the phase-locking feature of the temporal fine structure (TFS). Subsequently, a set of spectral-temporal response fields (STRFs) is applied to each pair of neurograms using a spectral-temporal Gabor function (Chi et al., 1999) derived as a function of crest frequency and drift velocity. Given the range of temporal modulation rates (2-32 Hz) and of spectral modulation scales (0.3-8 cycles/octave), STRFs simulate how the human auditory cortex encodes different sound features, adapts its response to temporal variations in the speech signal, captures the rhythmic and rate variations that are important components of time-domain modulation, and also captures the spectral information in speech through its sensitivity to a specific range of frequencies. The STRFs act as a cortical spectral-time domain modulation filter, simulating the way the human auditory cortex analyzes speech sounds in the spectral-temporal domain.

The given process generates four-dimensional complex-valued matrices with dimensions of scale (cycles per octave), rate (Hz), time (seconds), and frequency (Hz).

The complex-valued elements in each matrix were calculated before the cortical differences are computed (Wirtzfeld et al., 2017). The T token is a four-dimensional real-valued entity. It is obtained by subtracting the cortical response of the clean speech matched-noise signal from the cortical response of the clean speech signal. The N token is computed in the same manner using the chimaeric speech signal. The associated matched-noise representations are the same processing that was used to generate the matched-noise signal for the creation of speech chimaeras. So the STMI is calculated as

$$\text{STMI} = 1 - \frac{||T - N||^2}{||T||^2} \tag{3.1.1}$$

where $\|\cdot\|$ is the Euclidean-norm operator. For a matrix X with n elements indexed by k, T token represents the cortical response to a clean speech signal, and N token represents the cortical response for the associated chimaeric speech signal.

The STMI algorithm aims to quantify the difference in spectro-temporal modulation between a clean speech signal and its corresponding chimaera speech signal in order to assess the intelligibility of the speech signal. Focus on the value of STMI, which theoretically ranges from 0 to 1, the higher the value, the higher the intelligibility of the speech signal.

3.1.2 NSIM

The NSIM quantifies differences in neural spectro-temporal features using an image-based processing model (Hines and Harte 2010, 2012). As shown in Figure 3.1, NSIM works by comparing the clean speech neurogram (R) and a corresponding chimeric speech neurogram (D).

In an auditory model of the AN fiber response, mean rate and spike timing information

coexist in the same PSTH. This thesis first clarifies the mean rate information and the spike timing information. The spectral features supported by neural rate position coding and the temporal fluctuations of these features at rates below approximately 78 Hz are referred to as mean rate information, while the temporal fluctuations of neural firings at rates above 78 Hz and the precise time at which spikes occur due to acoustic transients are referred to as spike-timing information.

Therefore, in this thesis, the ENV-related neurogram and NSIM measurements of Hines and Harte (2010, 2012) will be referred to as the mean-rate (MR) neurogram and NSIMs, and since the TFS-related neurogram and NSIM measurements of Hines and Harte (2010, 2012) convey both mean-rate and spike-timing information, we will refer to them as the fine-timing (FT) neurogram and NSIMs.

On this basis, this thesis processes clean and chimaera speech neurograms to produce neurograms that reflect the corresponding cues for each information source. The mean-rate neurograms are generated by resampling the AN fiber PSTH responses to 100-µs bins in the consonant-vowel-consonant (CVC) target word regions of the unmodified neurograms and convolving them witha a 128-sample Hamming window. While the fine-timing neurograms are generated by retaining the 10-microsecond box sizes derived from the auditory periphery model and convolving them with a 32-sample Hamming window for each PSTH.

Since the NSIM focuses on the images over a windowed area rather than just using a simple point-to-point pixel comparison, the optimal statistics are computed within a 3×3 square window for both MR and FT neurograms (Hines and Harte 2010, 2012). Thus, three factors are measured for each individual pixel comparison: luminance, contrast, and structure. Luminance represents a comparison of the mean values of

neurograms. Contrast is a measure of the variance of neurograms, and structure is equivalent to the correlation coefficient between the neurograms.

Hines and Harte (2012) investigated the effect of weighting factors (α, β, γ) on phoneme discrimination in CVC word lists and found that the 'contrast' term (β) had little or no effect on overall NSIM performance. When the 'brightness' (α) and 'structure' (γ) terms were set to 1 and the 'contrast' (β) term was set to 0, they found that the results produced under these conditions had comparable accuracy and reliability to those calculated using the optimized values, so NSIM was simplified using this set of weighting factors.

A schematic of NSIM generation is also shown in Figure 3.1. In the figure, R represents for a clean speech neurogram, while D represents for an associated chimaeric speech neurogram. The equation for the NSIM is

$$\mathrm{NSIM}(R,D) = \left(\frac{2\mu_R\mu_D + C_1}{\mu_R^2 + \mu_D^2 + C_1}\right)^{\alpha} \cdot \left(\frac{2\sigma_R\sigma_D + C_2}{\sigma_R^2 + \sigma_D^2 + C_2}\right)^{\beta} \cdot \left(\frac{\sigma_{RD} + C_3}{\sigma_R\sigma_D + C_3}\right)^{\gamma} \tag{3.1.2}$$

where C1, C2, and C3 are three constants, which are regularization coefficients that prevent numerical instability (Wang et al. 2004). These three exponents are weighting coefficients, adjusting the relative proportions of the three components.

3.1.3 Cross-Correlation Coefficients

The neural Cross-Correlation Coefficients (CCC) measures the similarity between the envelope (ENV) or temporal fine structure (TFS) responses of clean speech signal and chimaeric speech signal (Heinz and Swaminathan, 2009). To obtain neural Cross-Correlation Coefficients, first, this thesis computes shuffled auto-correlograms (SACs), which are the shuffled all-order interval histograms calculated by counting

all intervals from all pairs of time for a single speech signal, and they are normalized by N(N-1)r$^2\triangle\tau$D, where N represents for all possible pairs of stimulus repetitions within a given set of spike trains, i.e., for N repetitions, there are $N \times (N-1)$ pairs. $\triangle\tau$ is the binwidth, r is the average discharge rate and D is the duration of the response window to allow a more intuitive interpretation of temporal coding. With this normalization, the baseline value is corrected to 1. Figure 3.2 illustrates some examples of SACs. Then the shuffled cross-correlograms (SCCs) are computed by the shuffled all-order interval histogram based on all possible pairs between two separate sets of speech signal or the cross-polarity speech signals. And they are normalized by $N_A N_B r_A r_B \triangle\tau$D. All variables shown in this equation is calculated between two separate sets of spike trains in response to conditions A and B.

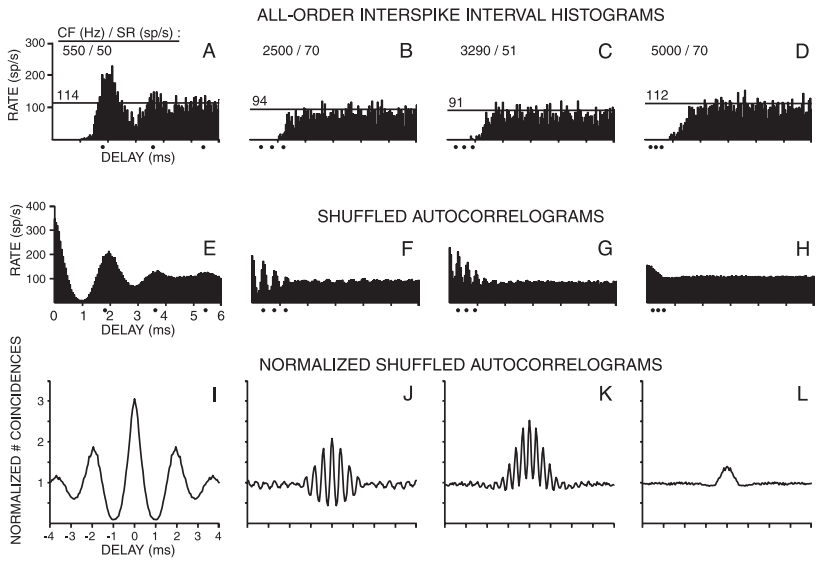

Figure 3.2: Examples of all-order interspike interval histograms (top), SACs (middle), and normalized SACs (bottom) for 4 auditory nerve (AN) fibers (4 columns). Reprinted from Fig. 3 of from Louage et al (2004).

Therefore, in this way, this thesis develops different cross-correlograms (SCCs). In

general, the SCC is computed by cross-polarity correlogram, that means obtaining the response to the original stimuli(A+) and its inverse polarity pair (A-), and the result is SCC(A+, A-). Then comes the second part, the cross-stimulus for SAC is obtained by averaging the SCC with different polarity stimuli pair of clean speech signal and chimaeric speech signal. The equation is

$$\mathrm{SAC_{AB}} = \frac{\mathrm{SCC(A+, B+)} + \mathrm{SCC(A-, B-)}}{2} \tag{3.1.3}$$

At last, the cross-stimulus cross-polarity correlogram compares the clean speech signal with inverse chimaeric speech signal, averaged with the inverse clean speech signal with chimaeric speech signal, the equation shows as follows

$$\mathrm{SCC_{AB}} = \frac{\mathrm{SCC(A+, B-)} + \mathrm{SCC(A-, B+)}}{2} \tag{3.1.4}$$

The next step is to produce the difcors and sumcors using shuffled correlograms. Difcors are computed as the difference between the shuffled auto-correlograms and the shuffled cross-correlograms with same stimulus, whereas sumcors were computed as the average of shuffled auto-correlograms and the shuffled cross-correlograms with same stimulus. It is worth noting that with respect to the sumcors, since only neural coding of phonemic ENV information is considered, a low-pass filter is designed in this thesis to limit the sumcors spectra to contain only frequencies below 64 Hz. (Swaminathan and Heinz, 2012).

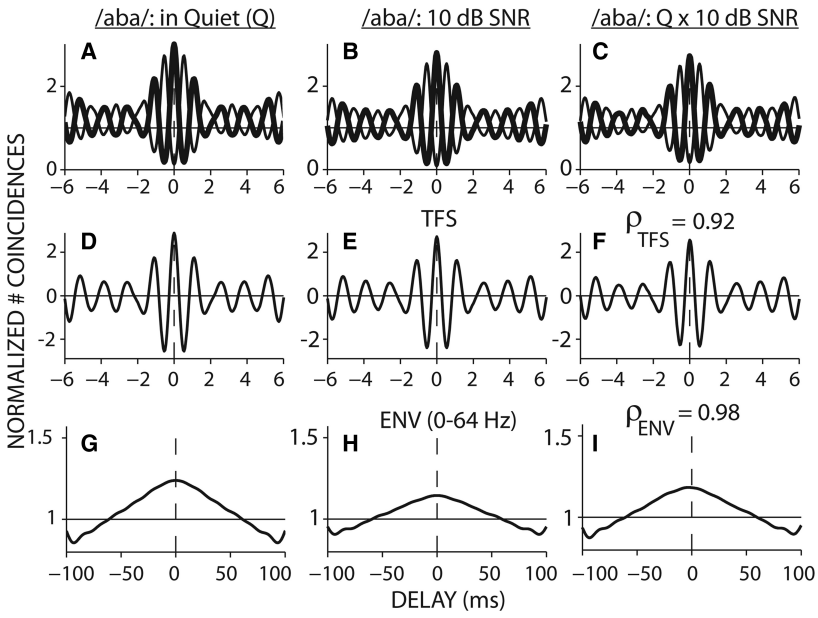

Figure 3.3: Correlogram analyses to quantify the neural coding of ENV and TFS in noise-degraded speech. Columns 1 and 2 show temporal coding in quiet and for a 10dB SNR, respectively. column 3 illustrates the similarity in temporal coding between these two conditions. A-B, Normalized shuffled auto correlograms (thick line) and cross-polarity correlograms (thin line). C, Shuffled cross-stimulus correlogram (thick line) and cross-polarity, cross-stimulus correlogram (thin line). D–F, Difcors represent TFS coding. G–I, Sumcors represent phonemic (0–64 Hz) ENV coding. Reprinted from Fig. 1 of Swaminathan & Heinz (2012).

More generally, shuffled correlograms quantify cross-correlation as a function of delay and demonstrate a peak at the characteristic delay (CD) between the two responses, and the CD is taken as zero when a single fiber is responding to two stimuli without a defined delay between one another (Heinz and Swaminathan, 2009). Figure 3.3 illustrates some examples of sumcors and difcors. Thus, on the basis of difcors and sumcors, the cross-correlation coefficient for temporal fine structure is calculated as:

$$\rho_{\mathrm{TFS}} = \frac{\mathrm{difcor_{AB}}}{\sqrt{\mathrm{difcor_A} \times \mathrm{difcor_B}}} \tag{3.1.5}$$

where all the difcors are evaluated at the corresponding CD, just as described above, the CD is the delay at which the difcor peak occurs, so the value is computed from the dicor peak height. Likewise, the cross-correlation coefficient for envelope is calculated from the corrected sumcors as:

$$\rho_{\mathrm{ENV}} = \frac{\mathrm{sumcor}_{\mathrm{AB}} - 1}{\sqrt{(\mathrm{sumcor}_{\mathrm{A}} - 1) \times (\mathrm{sumcor}_{\mathrm{B}} - 1)}} \tag{3.1.6}$$

where the sumcor value is computed from the sumcor peak height (after subtracting the baseline value of 1).

In this thesis, the range of CF is chosen to be 200 Hz-8 kHz and the number of AN fibers in each CF is 10. Choose all CFs $\leq$ 2.5 kHz for TFS (Johnson, 1980) and all CFs $\leq$ 8 kHz for ENV. The range of neural Cross-Correlation Coefficients is from 0 to 1, where 0 represents no correlation and the larger the value, the higher the correlation between the clean speech signal and chimaeric speech signal.

To properly understand the perceptual salience of TFS cues, it is important to consider that acoustic TFS generates recovered envelope cues as well as true neural TFS cues, just as shown in Figure 3.4. The results of the present study based on ρ_{ENV} provide physiological evidence for recovered envelope cues in response to AN to speech-noise chimeras (with noisy true envelope cues produced by cochlear filtering). Model predictions confirm that recovery envelopes are also present in TFS speech (without true envelope cues) and speech-speech chimeras (Swaminathan and Heinz, 2012). This suggests that mean-rate neural cues from the original ENV and the recovered ENV can explain some of the variability in subjective perceptual scores.

A brief visualization of how the speech signal transfers to neural signal. Acoustic

TFS and Acoustic ENV are referred to as TFS and ENV, and Neural TFS and Neural ENV are Fine-timing and Mean-rate for this study.

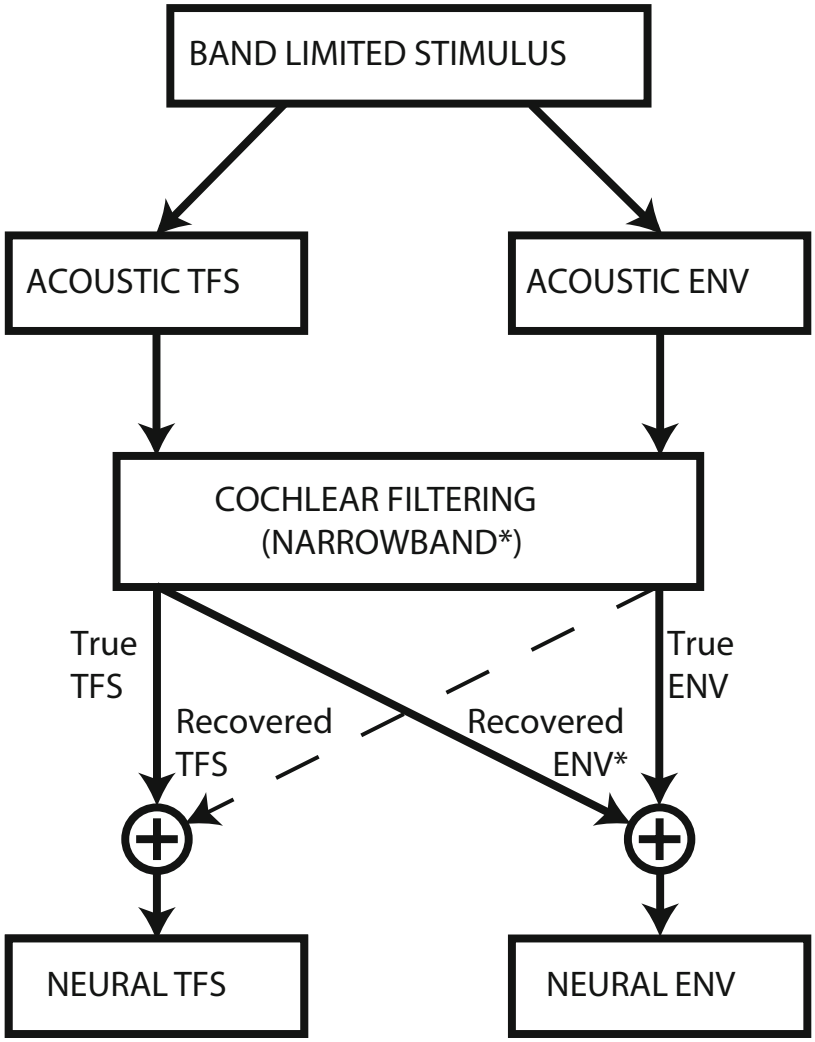

Figure 3.4: Difference between TFS and ENV. Reprinted from Fig. 10 of from Heinz & Swaminathan (2009).

3.1.4 Modulation to test Cross-Correlation Coefficients

Since the third method of predicting speech intelligibility, Cross-Correlation Coefficients, is more complex and involves more variables, the purpose of this subsection is to verify that the correct matrix has been applied to calculate Cross-Correlation Coefficients correctly as described in Section 3.1.3.

According to the discussion on ENV and TFS in chapter 2.1, when changing the envelope information, the cross-correlation coefficient for temporal fine structure should not change. Conversely, when changing the carrier signal information, the cross-correlation coefficient for envelope should not change, either. Hence, in this chapter, two signals (Signal A & B) are modulated and the equations are shown below.

$$s(t) = A[1 + kx(t)]cos(\omega_c t) \tag{3.1.7}$$

The most common form of carrier signal is a sinusoid, e.g., Acos(ω_ct), where A is the carrier amplitude, ω_c is the carrier frequency, x(t) is used to modulate the amplitude of carrier signal. In this section, x(t) is also a sine function, giving a sinusoidally amplitude modulated (SAM) signal. The modulation index k was set to a value of 1, giving 100 % modulation. When changing the modulation frequency, ρ_{TFS} should not change and the envelope correlation ρ_{ENV} between the two signals is strongest when the modulation frequencies are equal. When changing the carrier frequency, ρ_{ENV} should not change and the temporal fine-structure correlation ρ_{TFS} between the two signals is strongest when the carrier frequencies are equal. If the above conditions are all satisfied, this thesis can be considered to have indeed calculated the Cross-Correlation Coefficients correctly.

3.2 Results

According to the method described in Section 2.2.1 for generating speech chimaeras, the noise is chosen to be white Gaussian noise (WGN) or matched-noise (MN), respectively, and therefore five different chimaera signals are used in this thesis, which are ENV with MN TFS, Speech ENV with WGN TFS, Speech TFS with EGN ENV, Speech TFS with MN ENV, Speech TFS with MN ENV, and Speech TFS only (Speech TFS with flat ENV). The number of vocoder filters varies from 1, 2, 3, 6, 8, 16 to 32. In order to make the predictor value dependent only on the target word and not the proceeding phrase "Say the word", the thesis selects only the neurograms windowed around the target word for processing. In the following section, the results of the different methods are presented, and then analyzed and compared with regression models.

Following the methods described in Section 3.1, the results obtained for assessing speech intelligibility are presented in this section in sequence.

3.2.1 STMI Evaluation of Speech Chimaeras Intelligibility

Firstly, regarding the STMI method, Figure 3.5 shows the average STMI values versus the number of vocoder filters for the Speech ENV and Speech TFS chimaeras. For the Speech ENV chimaeras, as the number of vocoder filters grows, the STMI values show a clear increasing trend for different noise cases, from about 0.4 at the beginning to more than 0.8. And compared with the MN TFS, the WGN TFS shows an even more obvious increase, from the initial gap of nearly 0.1 with MN to the final point of almost the same.

For the Speech TFS chimaeras, as the number of vocoder filters grows, the STMI values basically show a decreasing trend for different noise cases. The difference is that the MN ENV decreases from around 0.36 to 0.21, and the trend is more stable and does not decrease significantly when the number of vocoder filters is less than 8, and starts to decrease to 0.21 when the number of filters is more than 8. On the other hand, the WGN ENV or Flat ENV, maintains a decreasing trend from around 0.55 to around 0.18, and When the number of vocoder filters is greater than 8, the fold is steeper and the decreasing trend is accelerated, but eventually the fall point for the different noise cases (number of vocoder filters is 32) is very close to each other, both are around 0.2.

Figure 3.6 shows the corresponding results from Wirtzfeld et al (2017), using the older version of the auditory periphery model of Zilany et al. (2009). Similar to that paper, this thesis also used the STMI method to estimate the speech intelligibility of five different speech chimaeras based on the new auditory periphery model of of Bruce et al. (2018).

Comparing Figure 3.5 with Figure 3.6, the Speech ENV chimaeras mixed with MN TFS and WGN TFS show better results as the number of vocoders increases, while the Speech TFS chimaeras accuracy decreases as the number of vocoders increases, with the slight difference that the Speech TFS chimaeras mixed with WGN ENV and Flat ENV show a more significant decreasing trend of the results obtained in this thesis when the number of vocoders is less than 6. It can be seen that the results obtained from the two figures are very close to each other and the differences in the results can be attributed to the differences in the auditory periphery models.

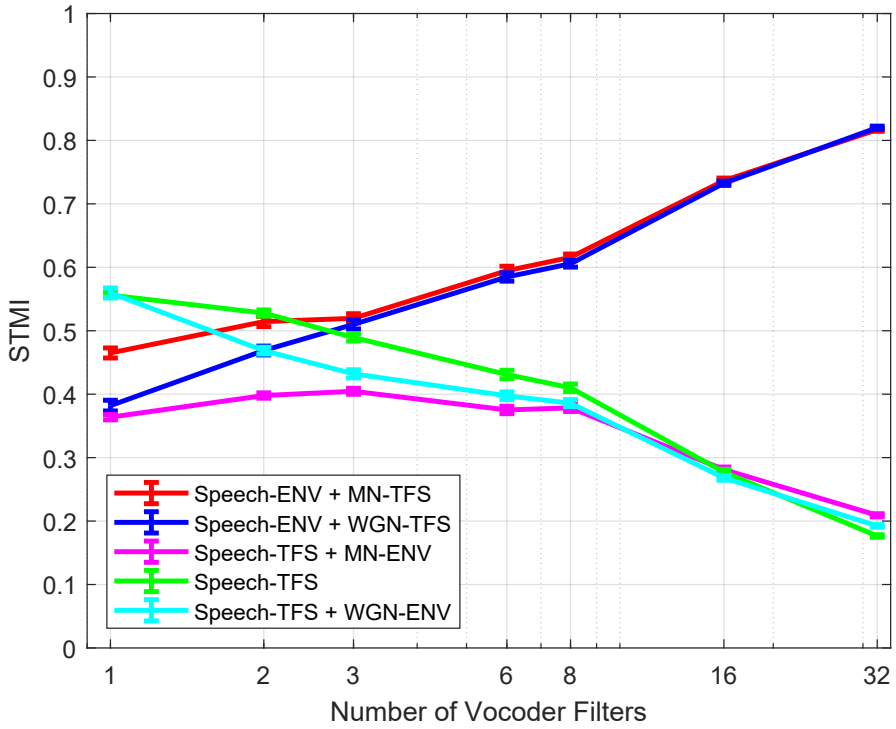

Figure 3.5: Average STMI values (error bars ± 1 SEM) as a function of the number of vocoder filters.

3.2.2 NSIM Evaluation of Speech Chimaeras Intelligibility

For the NSIM method, Figure 3.7 shows the average mean-rate NSIM values versus the number of vocoder filters for the Speech ENV and Speech TFS chimaeras. For the Speech ENV chimaeras, as the number of vocoder filters grows, the mean-rate NSIM values show an upward trend for the different noise scenarios, from around 0.35 initially to close to 0.6. Compared to the MN TFS, the WGN TFS shows more of an increase, but the trend is very similar.

For the Speech TFS chimaeras, as the number of vocoder filters grows, the mean-rate

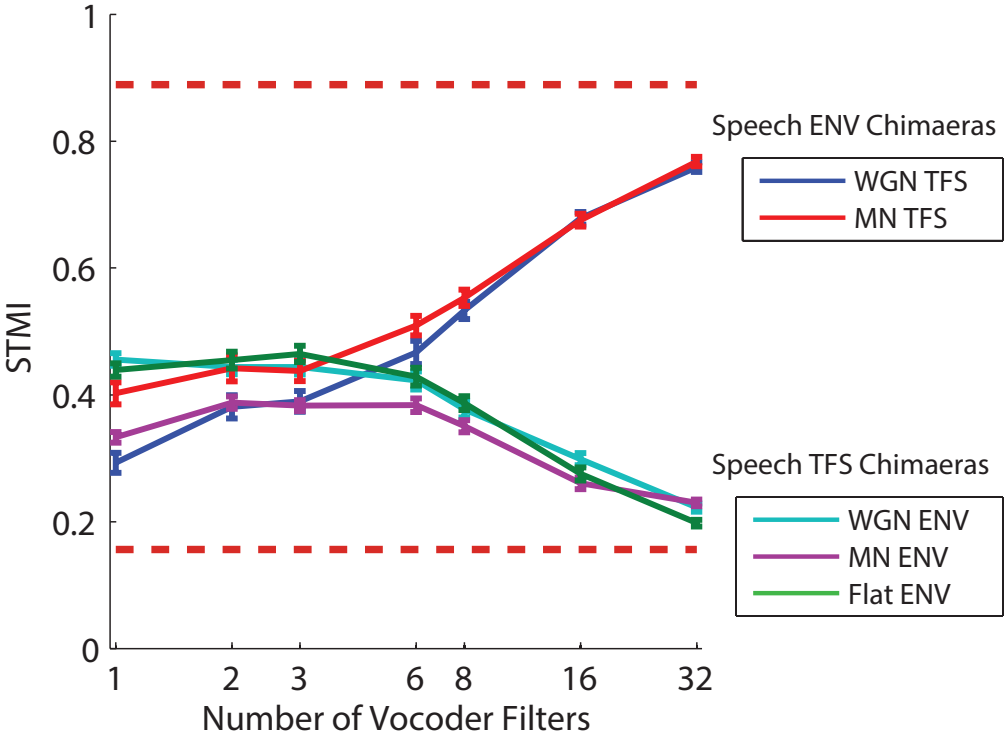

Figure 3.6: Average STMI values (error bars ±1 SEM) as a function of the number of vocoder filters. The horizontal dashed lines in each panel show the empirically determined lower and upper metric bounds. The lower bound is 0.16 and the upper bound is 0.89. Reprinted from Fig. 5a of Wirtzfeld et al (2017).

NSIM values do not change significantly for the different noise cases.The MN ENV rises from initially around 0.37 to 0.45, that is, when the number of vocoder filters is 8, but the number greater than The Flat ENV, although fluctuates a bit, the mean-rate NSIM values do not change significantly, numerically going from 0.44 to 0.49 and then back to 0.44. Compared to the Flat ENV, the WGN ENV has a more obvious decreasing trend from the initial 0.44 when the number of vocoder filters is 1 to the final 0.38

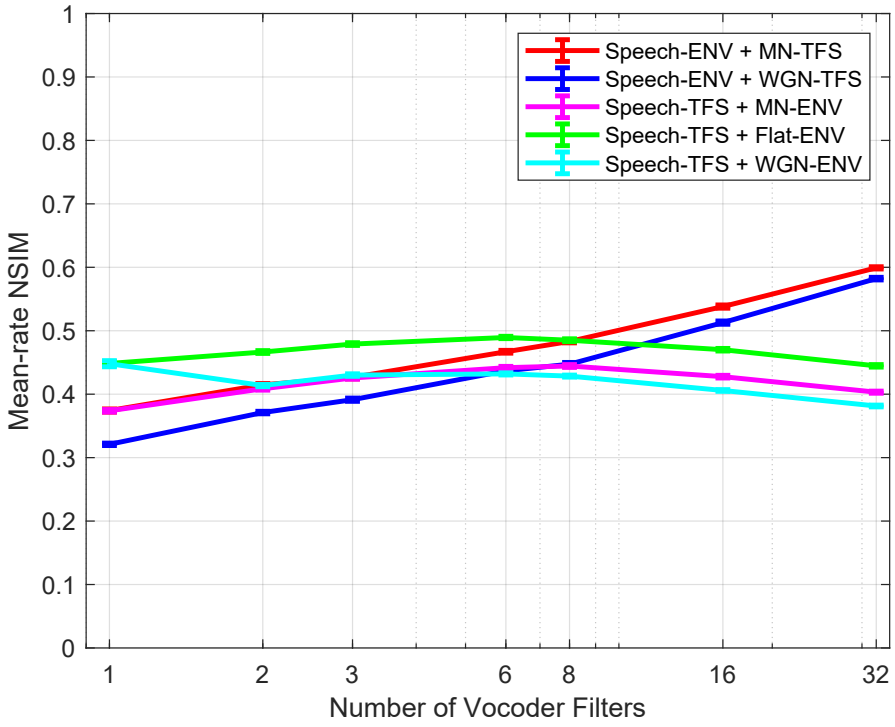

Figure 3.7: Average mean-rate NSIM values (error bars ± 1 SEM) as a function of the number of vocoder filters.

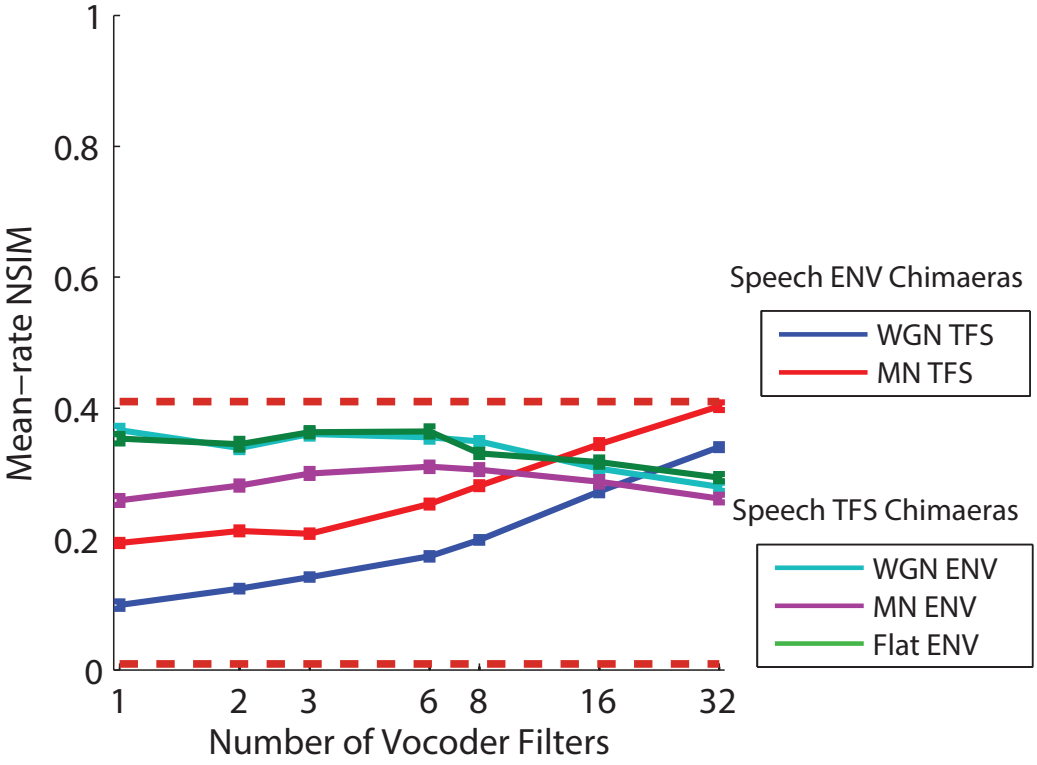

Figure 3.8: Average mean-rate NSIM values as a function of the number of vocoder filters. The horizontal dashed lines show the empirically determined lower and upper bounds. The lower bound is 0.0090 and the upper bound is 0.41. Reprinted from Fig. 6a of Wirtzfeld et al (2017).

Figure 3.8 is also from Wirtzfeld et al (2017). Similarly, in this thesis, the NSIM method is used to detect speech intelligibility for five different speech chimeras. Comparing Figure 3.7 and Figure 3.8, the values of the Speech ENV chimaeras mixed with MN TFS and WGN TFS increases as the number of vocoders increases, while the values of the Speech TFS chimaeras shows a decrease in a fluctuating state. It can be seen that the trends of the polylines of both figures are very similar.

For the NSIM method, Figure 3.9 shows the average fine-timing NSIM values versus the number of vocoder filters for the Speech ENV and Speech TFS chimaeras. For

the Speech ENV chimaeras, as the number of vocoder filters grows, the fine-timing NSIM values do not change obviously for different noise cases, and the curve in the figure is flat, from around 0.04 at the beginning to around 0.06 at the end.

For the Speech TFS chimaeras, as the number of vocoder filters grows, fine-timing NSIM values also do not change obviously for different noise cases. In contrast, MN ENV increases slightly more, from around 0.15 to 0.21, while WGN ENV or Flat ENV, the curve is more flat, with no significant change in the values, except that the difference between WGN ENV and Flat ENV is slightly larger, from almost the same to 0.03 when number of vocoder filters of 32.

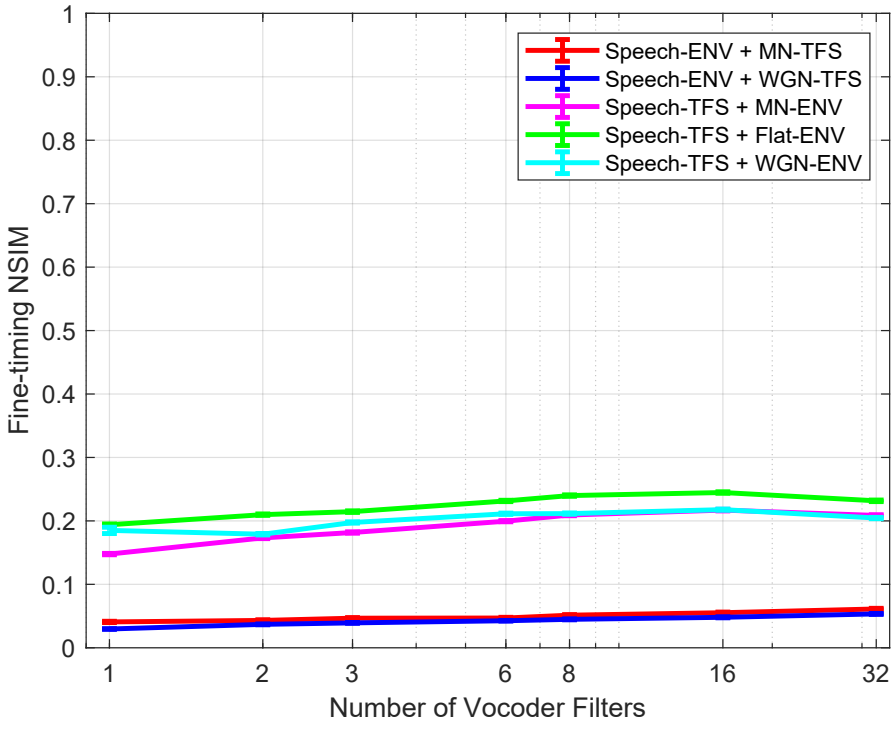

Figure 3.9: Average fine-timing NSIM values (error bars ± 1 SEM) as a function of the number of vocoder filters.

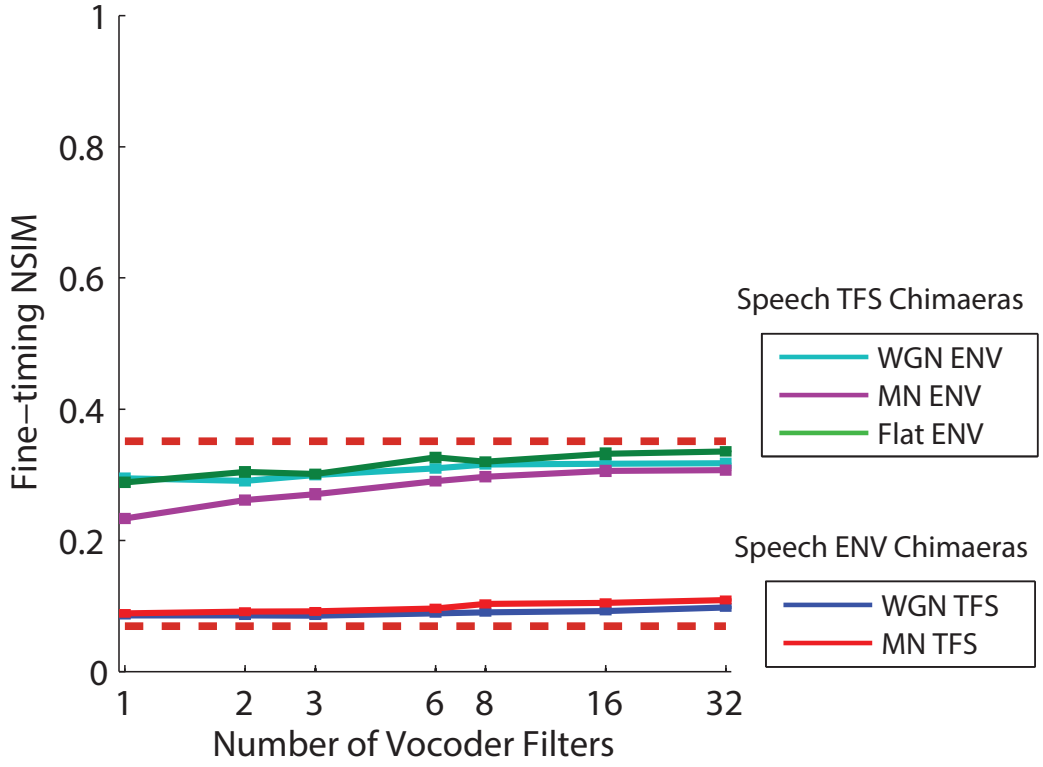

Figure 3.10: Average fine-timing NSIM values as a function of the number of vocoder filters. The horizontal dashed lines show the empirically determined lower and upper bounds. The lower bound is 0.069 and the upper bound is 0.35. Reprinted from Fig. 7a of Wirtzfeld et al (2017).

Figure 3.10 also from Wirtzfeld et al (2017). Comparing Figures 3.9 and 3.10, the Speech ENV chimaeras values remains almost unchanged at a low level as the number of vocoders increases. While the values of the Speech TFS chimaeras has a little increase. It can be seen that the trends of the polylines in the two figures are very similar as well.

3.2.3 Cross-Correlation Coefficients Evaluation of Speech Chimaeras Intelligibility

This thesis begins by verifying the correctness of the adopted Cross-Correlation Coefficients (CCC) algorithm, as described in Section 3.1.4, two signals A and B are generated, and the modulation frequency of signal A is set to 10 Hz, and the modulation frequency of signal B is varied so that the test interval is 2-20 Hz, resulting in Figure 3.11 (a), and it shows the average Cross-Correlation Coefficients ENV values versus the modulation frequency of test signal. From the figure, it can be seen that as the modulation frequency increases, the Cross-Correlation Coefficients ENV values peak at 10Hz and remain around 0.2 at other frequencies, except for the increase of the terminal.

Figure 3.11 (b) shows the average Cross-Correlation Coefficients TFS values versus the modulation frequency of test signal. From the figure, it can be seen that as the modulation frequency increases, the Cross-Correlation Coefficients TFS values produce a fluctuation as the modulation frequency increases, but the values are centered around 0.8.

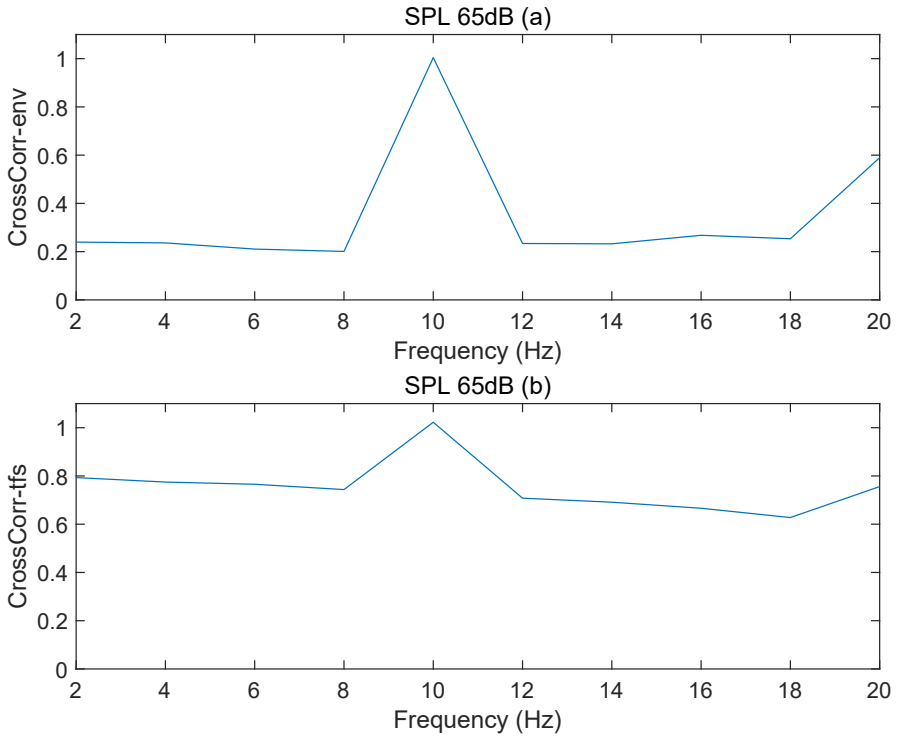

Figure 3.11: Mean Cross-Correlation Coefficients values versus the modulation frequency of test signal.

The next test is to change the carrier frequency, signal A carrier frequency is set to 1000 Hz, change the carrier frequency of signal B, the test interval is 800-1200 Hz, so Figure 3.12 (a) is obtained, and it shows the average CCC ENV values versus the carrier frequency. As can be seen from the figure, the CCC ENV values fluctuate as the carrier frequency increases, but the values are around 0.9. Compared with Figure 3.10(a), the fluctuation range in this figure is smaller and the simulation shows better performance.

And Figure 3.12(b) shows the average CCC TFS values versus the carrier frequency. From the figure, it can be seen that as the carrier frequency increases, the CCC TFS values peaks at 1000Hz and approaches zero at other frequencies.

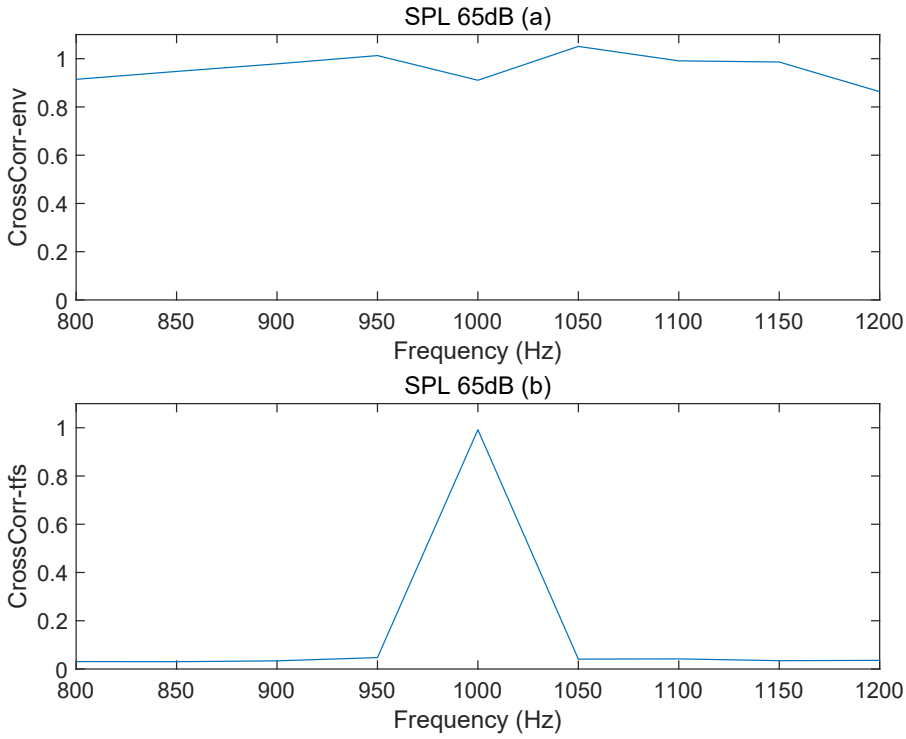

Figure 3.12: Mean Cross-Correlation Coefficients values versus the carrier frequency of test signal.

By testing, this thesis can be considered to have indeed calculated the Cross-Correlation Coefficients correctly. And the simulation effect of changing the carrier frequency is even better.

For the Cross-Correlation Coefficients method, Figure 3.13 shows the average Cross-Correlation Coefficients ENV values versus the number of vocoder filters for the Speech ENV and Speech TFS chimaeras. For the Speech ENV chimaeras, as the number of vocoder filters grows, the Cross-Correlation Coefficients ENV values show a clear increasing trend for different noise cases, with the curve increasing almost linearly from the initial around 0.53 to close to 0.87, and the increasing trend for the

MN TFS and WGN TFS is very similar.

For the Speech TFS chimaeras, as the number of vocoder filters grows, the Cross-Correlation Coefficients ENV values basically show a decreasing trend for different noise cases. The difference is that WGN ENV decreases more significantly, from the initial 0.51 to about 0.40, and when the number of vocoder filters is greater than 2, the trend is more stable and does not decrease significantly. On the other hand, the MN ENV has been decreasing from 0.44 to 0.39, and the Flat ENV has been decreasing similarly to the MN ENV from 0.47 to 0.42.

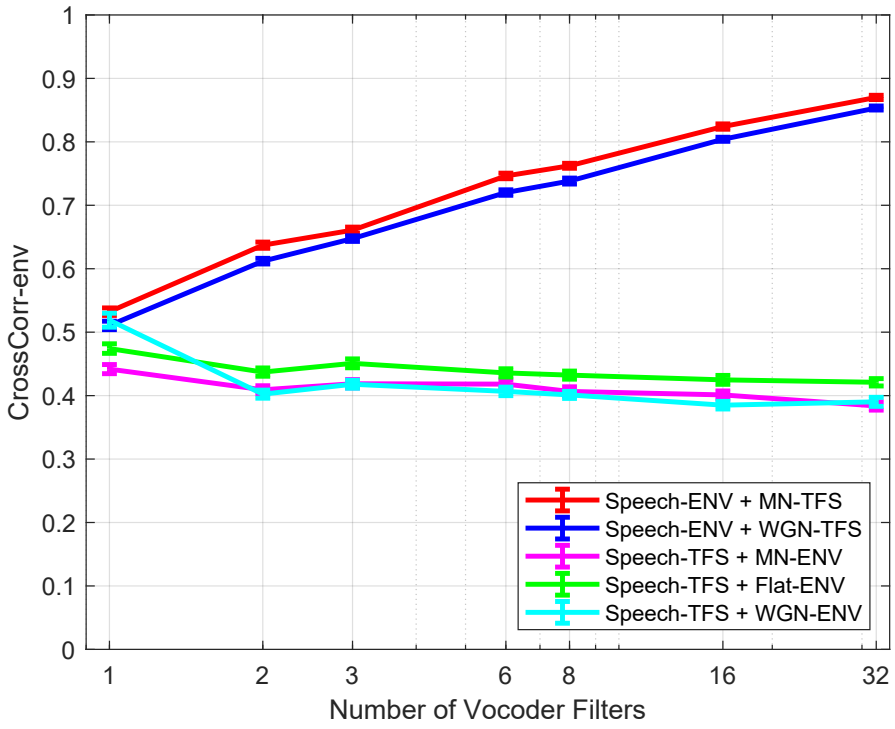

Figure 3.13: Mean Cross-Correlation Coefficients ENV values as a function of the number of vocoder filters. Mean alues across AN fibers [all eight CFs $\leq$ 2.5 kHz for TFS (Johnson, 1980); all 10 CFs $\leq$ 8 kHz for ENV] are plotted with SEM bars(error bars $\pm$ 1 SEM).

And Figure 3.14 shows the average Cross-Correlation Coefficients. TFS values versus the number of vocoder filters for the Speech ENV and Speech TFS chimaeras. For the Speech ENV chimaeras, as the number of vocoder filters grows, the cross-correlation coefficients. TFS values do not change significantly for different noise scenarios, and the curve in the figure is flat and stays around 0.3.

For the Speech TFS chimaeras, as the number of vocoder filters grows, the cross-correlation coefficients values show a slight increasing trend for different noise cases. In contrast, the MN ENV increases more, from around 0.49 to 0.62, while the WGN ENV or Flat ENV, both fluctuates and maintains its increase, except that the difference between the WGN ENV and the Flat ENV is slightly larger, from almost the same to 0.05 at the final when number of vocoder filters of 32.

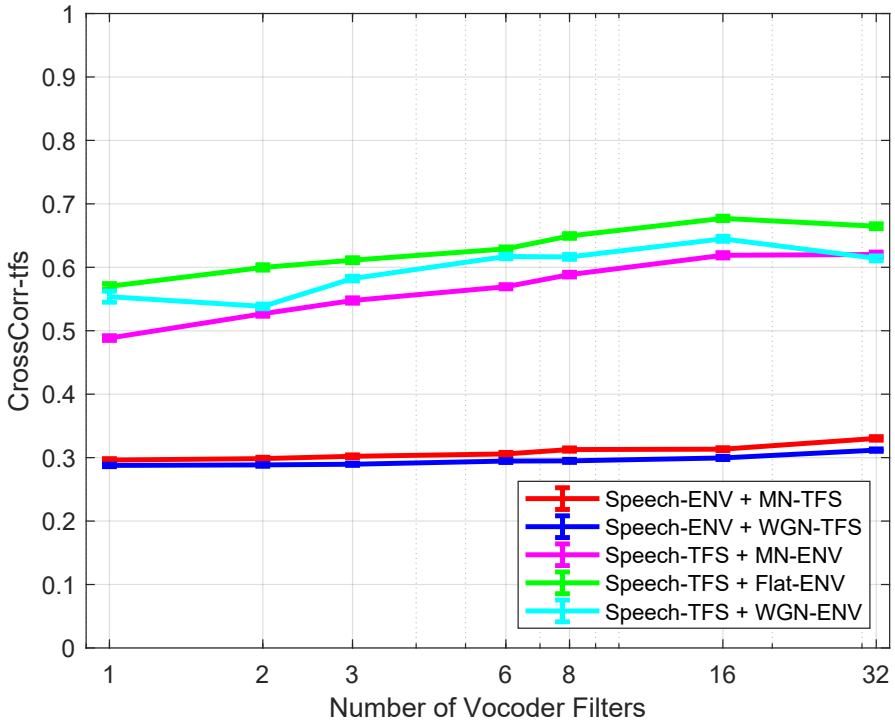

Figure 3.14: Mean Cross-Correlation Coefficients TFS values as a function of the number of vocoder filters. Mean values across AN fibers are plotted with SEM bars (error bars ± 1 SEM).

3.2.4 Correlations Between Predictions and Perceptions

As described in previous sections, this thesis uses three methods to predict speech intelligibility, which are Spectra-Temporal Modulation Index, Neurogram Similarity Index Measure, and Cross-Correlation Coefficients, respectively, and also obtains the corresponding prediction results. And the next step is to compare and summarize the effect of each method.

In this chapter, the establishment of a regression model between the predicted and perceptual values from the human speech intelligibility experiment of Wirtzfeld et al. (2017) is used for comparison. Since there are seven different numbers of vocoders, so for each chimaera signal there are seven data points, a total of five chimaera models, which means that each regression model uses 35 data points.

Several first-order linear regression models were constructed using the neural measures and the perceptual scores, using the general form of

$$\mathrm{RAU(PC)} = \mathrm{b0} + \mathrm{b1} \cdot M_1 + \mathrm{b2} \cdot M_2 + \mathrm{b3} \cdot M_1 \cdot M_2 \tag{3.2.1}$$

where RAU(PC) are the average rationalized arcsine transformed (RAU; Studebaker 1985) fractional phonemic-level scores for the CVC target-words, and M_1 and M_2 correspond to two neural measures (Wirtzfeld et al., 2017). The RAU transform is a method used in speech research to mitigate the floor and ceiling effects commonly observed in perceptual performance data.

For models with a single neural predictor, M_2 and M_3 are set to zero. For models with more than two neural measures, each measure has its own term and is combined with each of the remaining measures to form a two-term product interaction term. When p values are smaller than the set value, the larger the value of Adjusted R-Squared,

the better the performance. The Adjusted R-Squared is a metric utilized in statistics to assess the quality of fit of a linear regression model, representing an enhancement over the conventional R-Squared. The resulting perceptual identification figure is shown in Figure 3.15.

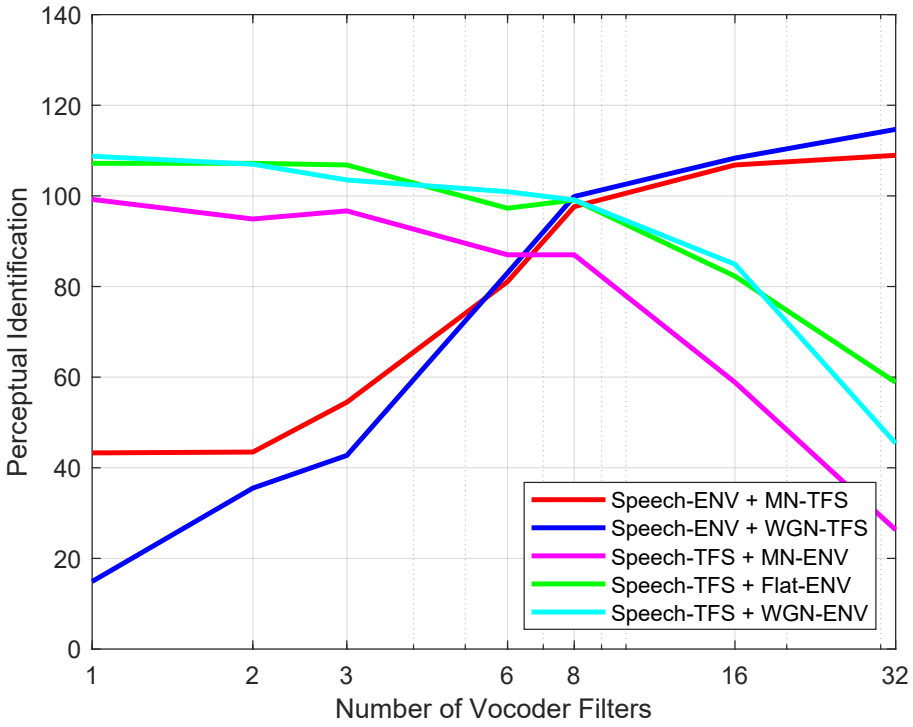

Figure 3.15: RAU-transformed perceptual measure identification values as a function of the number of vocoder filters.

The RAU-transformed perceptual measures are fitted to a linear regression model with the predicted values obtained from the STMI algorithm, and the variation of the obtained fitted values with the number of vocoders is shown in Figure 3.16.

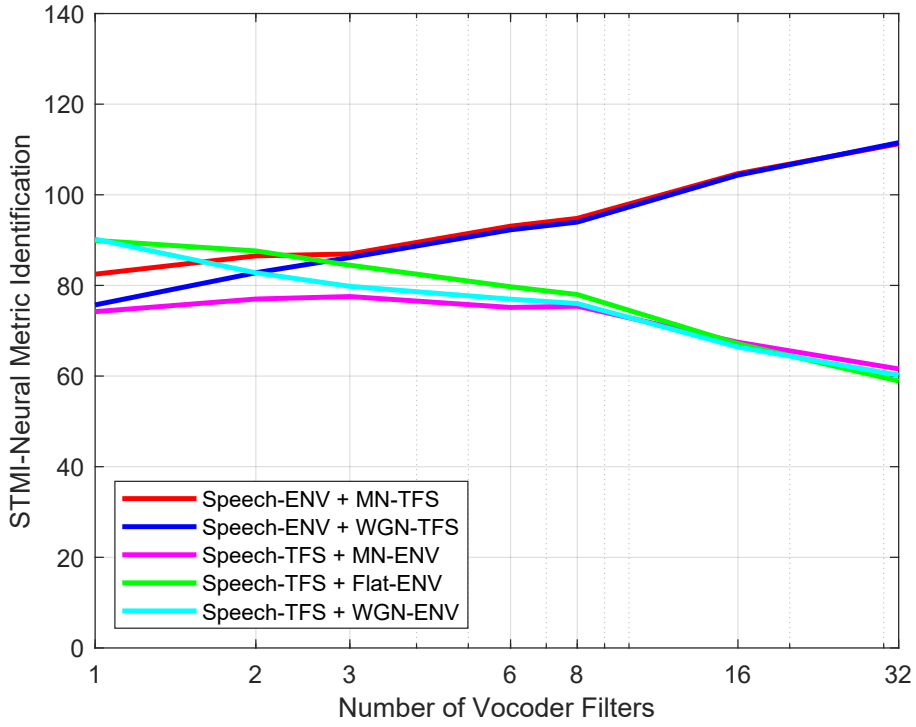

Figure 3.16: Predicted values obtained by STMI algorithm after fitting as a function of the number of vocoder filters.

Analyzing Figure 3.15 and Figure 3.16, this thesis finds that the trends of the corresponding polylines are basically the same, but the range of values varies a lot. Comparing these fitted values with the perceptual recognition values, the plot of actual measurements versus theoretical predictions is drawn as shown in Figure 3.17.

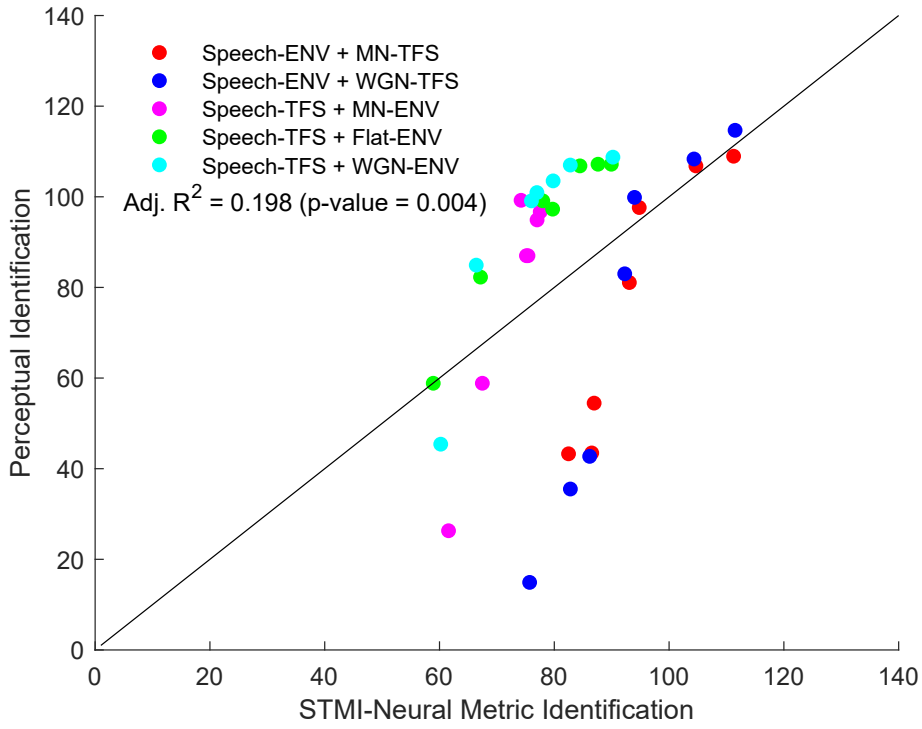

Figure 3.17: Fitting results of the predicted identification values obtained by the STMI algorithm with the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

The diagonal line represents a one-to-one correspondence between the perceptual values and the associated predictions. For points lying under the line, the model prediction is higher than the perceptual score, while for points above the line the prediction is lower. This paper uses the p-value to determine whether the data is significant, that is, whether it helps us decide whether the model is suitable for predicting speech intelligibility. As shown in the figure, the adjusted R^2 value is 0.198 (significant at p-value <0.05), which is not ideal, and careful inspection of the figure reveals that the STMI-based model overpredicts speech ENV chimeras and underpredicts speech TFS chimeras.

The perceptual measures are fitted to a linear regression model with the predicted values obtained from the NSIM algorithm, this thesis tries different models including the mean-rate NSIM and fine-timing NSIM measures independently, as well as their combination. Table 3.1 summarizes the adjusted R^2 value and p value obtained from different NSIM models.

Table 3.1: Linear regression models with NSIM predictions

MR NSIM		**FT NSIM**		**MR NSIM and FT NSIM**	
Adj.R^2	0.423	Adj.R^2	0.036	Adj.R^2	0.486
p value	<0.001	p value	0.140	p value	<0.001

As can be seen from the Table 3.1, using the mean-rate NSIM individually has a slightly higher the adjusted R^2 value of 0.423 (significant at p-value <0.001), while the p-value of fine-timing NSIM is larger than the set value, so its results are not significant, which means that fine-timing NSIM is not applicable for predicting speech chimeras individually.

The combination of the mean-rate NSIM and the fine-timing NSIM leads to improved predictions with an adjusted R^2 value of 0.486 (significant at p value <0.001). This provides cues for this thesis about combining information, and trying different combinations might improve prediction accuracy.

In order to improve the prediction results, this thesis tries to use the combination of different methods. The combination of STMI with MR NSIM and FT NSIM is

attempted separately and a distinction is made between the presence and absence of interactions in combination. Table 3.2 summarizes the adjusted R^2 value and p value obtained from the combination of STMI and NSIM predictions.

Table 3.2: Linear regression models with the combination of STMI and NSIM predictions

STMI and MR NSIM			**STMI and FT NSIM**		
Adj.R^2	0.467	with interaction	Adj.R^2	0.768	with interaction
p value	<0.001		p value	<0.001	
Adj.R^2	0.407	no interaction	Adj.R^2	0.776	no interaction
p value	<0.001		p value	<0.001	

As can be seen from the Table 3.2, STMI combined with MR NSIM has a small difference in accuracy compared to MR NSIM and FT NSIM combined, with a slight decrease in accuracy when the interaction is removed. While the accuracy of STMI combined with FT NSIM is significantly improved, the fitted R^2 value is 0.768 (significant at p-value <0.001), and the accuracy prediction is even better when there is no interaction. Plot the fitted predictions of the combination of STMI and FT NSIM and compare them to the perceptual measurements.

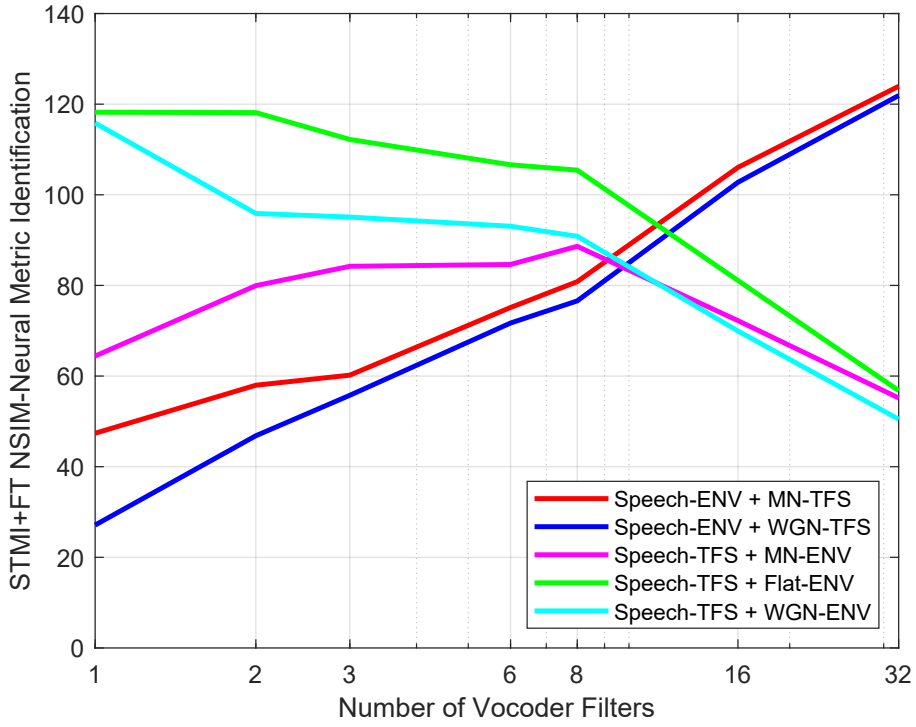

Figure 3.18: STMI combined with FT NSIM Predictions obtained after fitting considering the interaction case as a function of the number of vocoder filters.

In this thesis, a comparison of Figure 3.18 with the RAU-converted perceptual measurements (Figure 3.15) reveals that the trends of the polylines in the plots are basically the same, except for the Speech TFS with MN ENV, which corresponds to the purple line that basically shows a decreasing trend in perceptual measurements as the number of vocoders increases, while in the prediction figure, with the number of vocoders 8 as the turn, the front of this fold line shows an increasing trend instead, and there is a difference in the range of values.

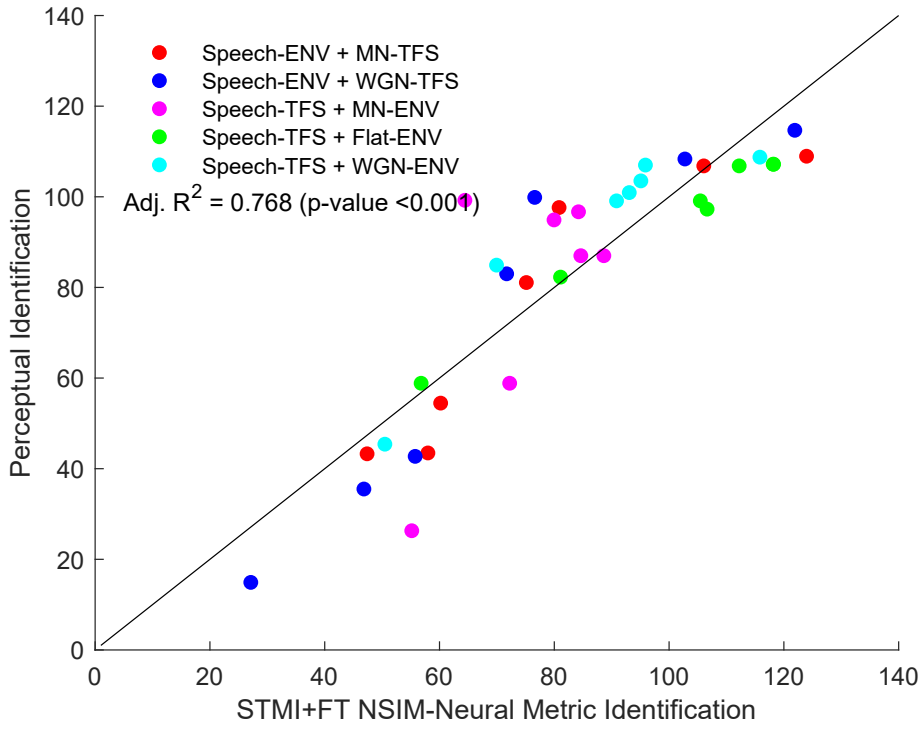

Figure 3.19: Fitting results of the predicted identification values obtained by STMI combined with FT NSIM (with interactions) and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

Observing the comparison of the predicted values and the perceptual measurements in Figure 3.19, the data points in the figure are almost around the diagonal line. Specifically, it slightly underestimates the Speech TFS with Flat ENV and more significantly overestimates the Speech TFS with MN ENV, but is generally closer to the diagonal.

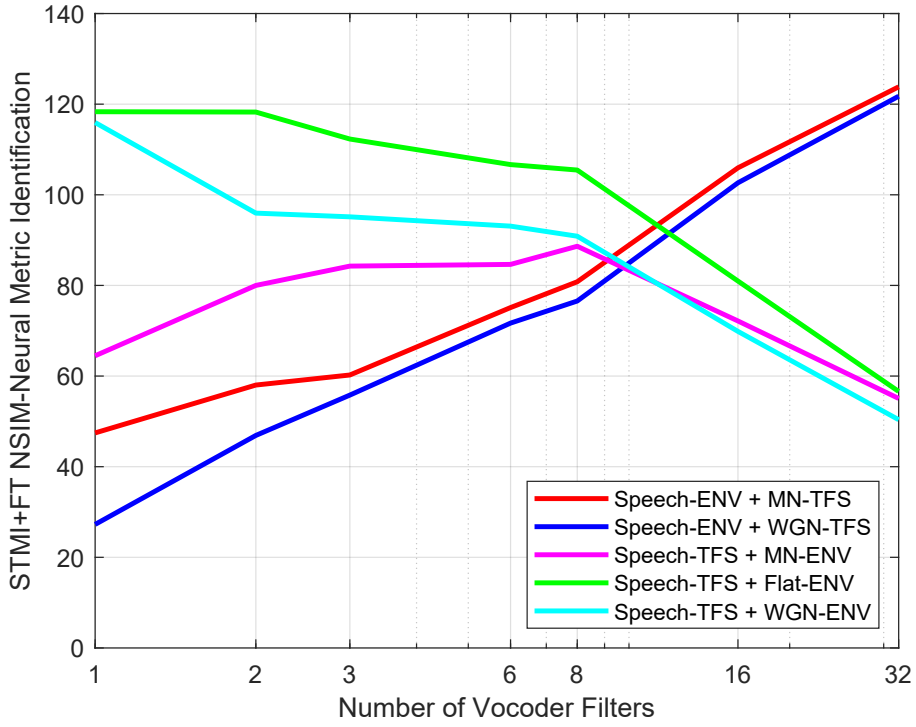

Figure 3.20: STMI combined with FT NSIM Predictions obtained after fitting ignoring interactions as a function of the number of vocoder filters.

Figure 3.20 is the results of STMI combined with FT NSIM Predictions obtained after fitting ignoring interactions, compared with the RAU transformed perceptual figure(Figure 3.15), this thesis finds that the trends of the polylines in the figure are basically the same, except for the Speech TFS with MN ENV, which has the similar issues as Figure 3.18, and there are differences in the range of values.

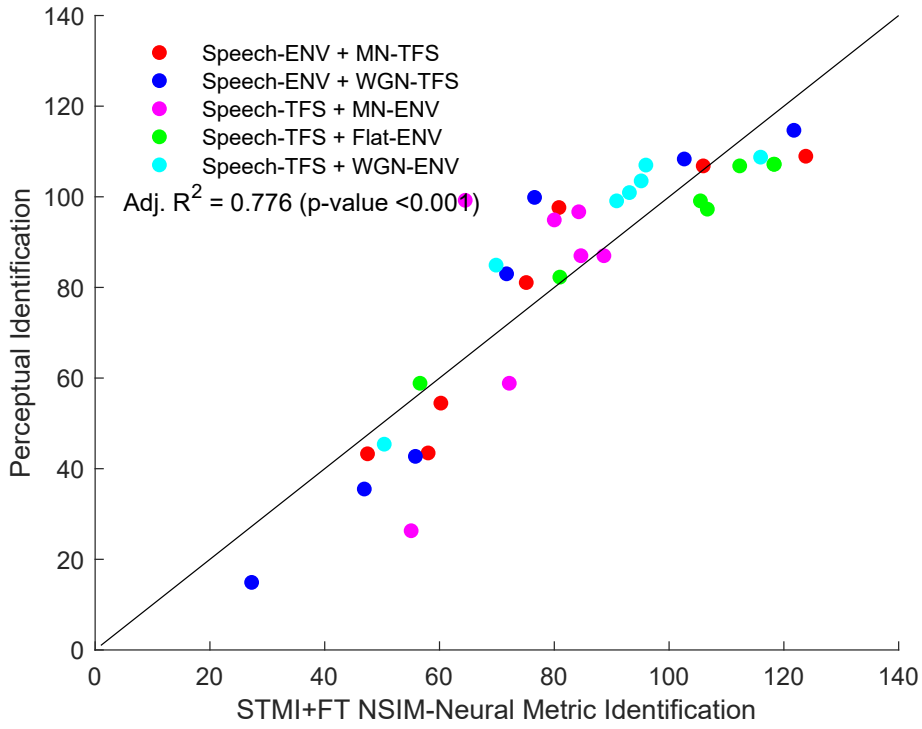

Figure 3.21: Fitting results of the predicted identification values obtained by STMI combined with FT NSIM (no interactions) and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

Observing the comparison of the predicted values and the perceptual measurements in Figure 3.21, similar to Figure 3.19, the data points in the figure are almost around the diagonal line, and the connecting lines of the data points are similar to the diagonal line, and the presence or absence of interactions does not have much effect, which indicates that the prediction results of the combination of STMI and FT NSIM are more in line with the perceptual measures, and that the algorithm is suitable for predicting the intelligibility of speech chimaeras.

This is followed by an evaluation of the third algorithm, Cross-Correlation Coefficients (CCC). Again, different models are tried in this thesis, including the individual CCC ENV and CCC TFS, as well as their combination. Table 3.3 summarizes the adjusted R^2 values and p-values derived from the different CCC models.

Table 3.3: Linear regression models with CCC predictions

CCC ENV		**CCC TFS**		**CCC ENV and CCC TFS**	
Adj.R^2	-0.0132	Adj.R^2	0.018	Adj.R^2	0.436
p value	0.461	p value	0.210	p value	<0.001

As can be seen from the Table 3.3, the p-value of CCC ENV and CCC TFS individual is larger than the set value, so its results are not significant, which means that neither CCC ENV nor CCC TFS is applicable for predicting speech chimaeras individually. As for the combination of the CCC ENV and CCC TFS, it leads to improved predictions with an adjusted R^2 value of 0.436 (significant at p value <0.001) compared to applying them individually. However, compared to previous combinations, this prediction accuracy is not as good as the combination of STMI and FT NISM, so this thesis will continue to try more combinations of Cross-Correlation Coefficients.

In this thesis, the combination of STMI with CCC ENV and CCC TFS is attempted separately and a distinction is made between the presence and absence of interactions in combination. Table 3.4 summarizes the adjusted R^2 value and p value obtained from the combination of STMI and CCC predictions.

Table 3.4: Linear regression models with the combination of STMI and NSIM predictions

STMI and CCC ENV			**STMI and CCC TFS**		
Adj.R^2	0.579	with interaction	Adj.R^2	0.759	with interaction
p value	<0.001		p value	<0.001	
Adj.R^2	0.507	no interaction	Adj.R^2	0.766	no interaction
p value	<0.001		p value	<0.001	

As can be seen from the Table 3.5, STMI combined with CCC ENV has better performance in accuracy compared to CCC ENV and CCC TFS combined, with a slight decrease in accuracy when the interaction is removed, but the accuracy is still not high. While the accuracy of STMI combined with CCC TFS is significantly improved, the fitted R^2 value is 0.759 (significant at p-value <0.001), and the accuracy prediction is even better when there is no interaction. Plot the fitted predictions of the combination of STMI and CCC TFS and compare them to the perceptual measurements.

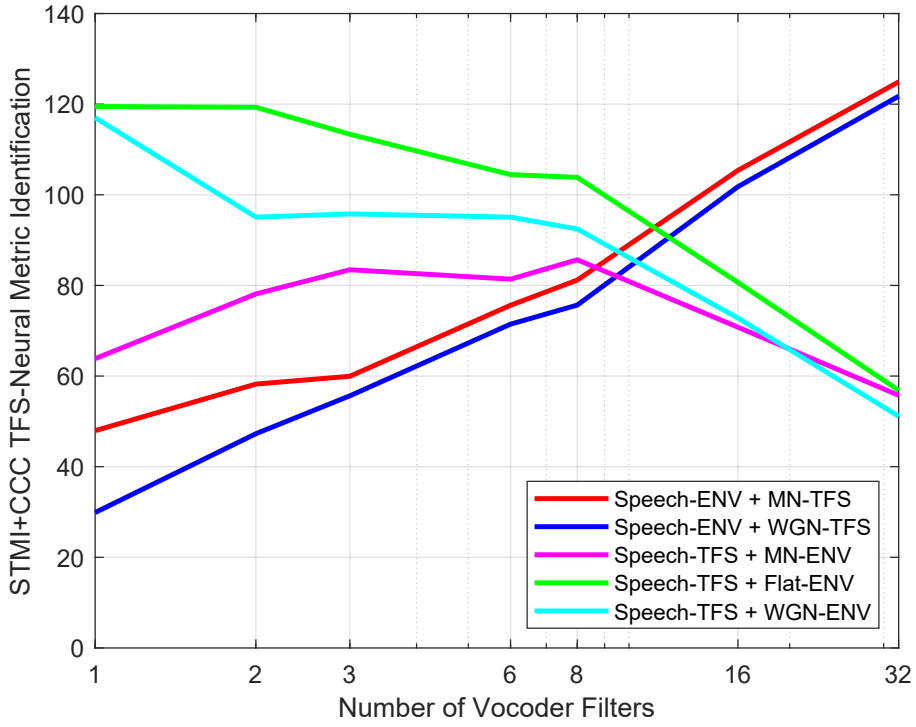

Figure 3.22: STMI combined with CCC TFS Predictions obtained after fitting considering the interaction case as a function of the number of vocoder filters.

Figure 3.22 is the results of STMI combined with CCC TFS Predictions obtained after fitting considering interactions, compared with the RAU transformed perceptual figure (Figure 3.15), this thesis finds that the trends of the polylines in the figure are basically the same, except for the Speech TFS with MN ENV, which has the similar issues as Figure 3.18, and there are still some differences in the range of values.

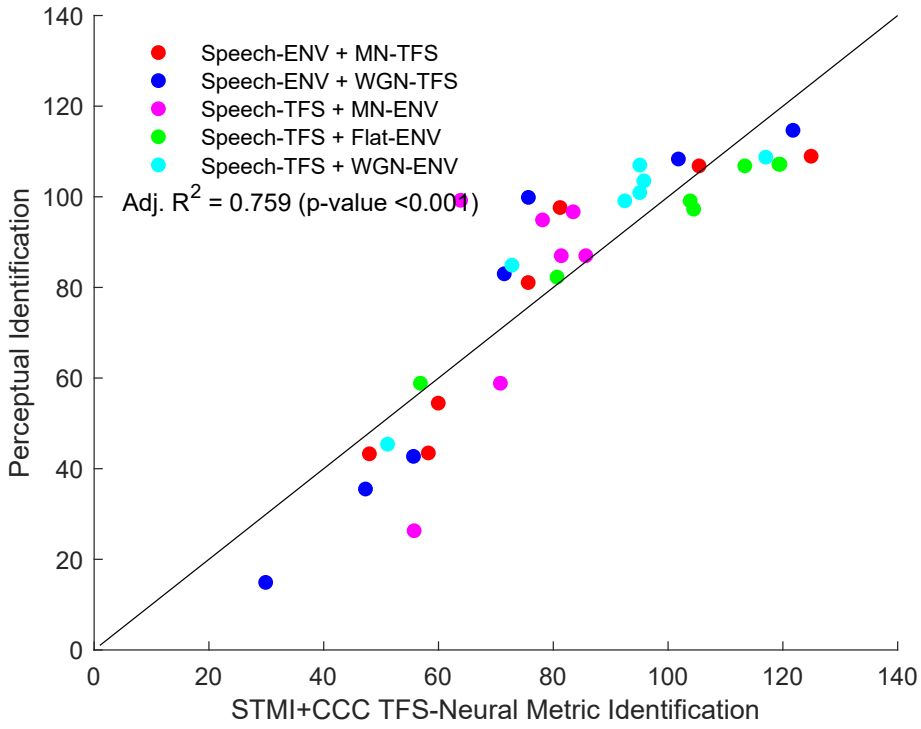

Figure 3.23: Fitting results of the predicted identification values obtained by STMI combined with CCC TFS (with interactions) and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

Observing the comparison of the predicted values with the perceptual measurements in Figure 3.23, the data points in the figure are almost all near the diagonal line. Specifically, it slightly underestimates the Speech TFS with Flat ENV and more significantly overestimates the Speech TFS with MN ENV, but is generally closer to the diagonal.

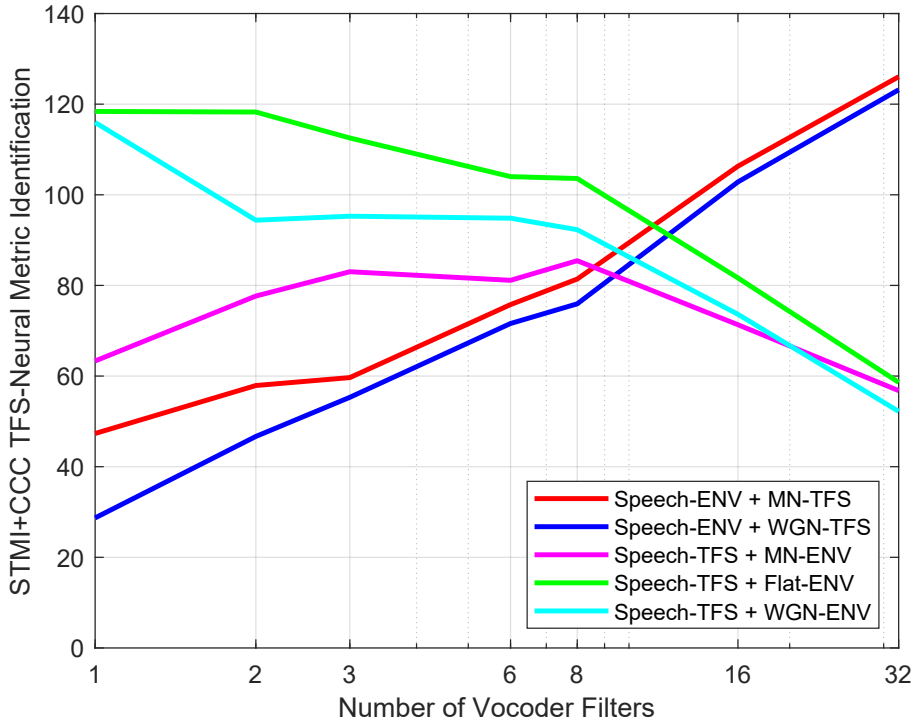

Figure 3.24: STMI combined with CCC TFS Predictions obtained after fitting ignoring the interaction case as a function of the number of vocoder filters.

Figure 3.24 is the results of STMI combined with CCC TFS Predictions obtained after fitting ignoring interactions, compared with the RAU transformed perceptual figure(Figure 3.15), this thesis finds that the trends of the polylines in the figure are basically the same, except for the Speech TFS with MN ENV, which has the similar issues as Figure 3.18, and there are still some differences in the range of values.

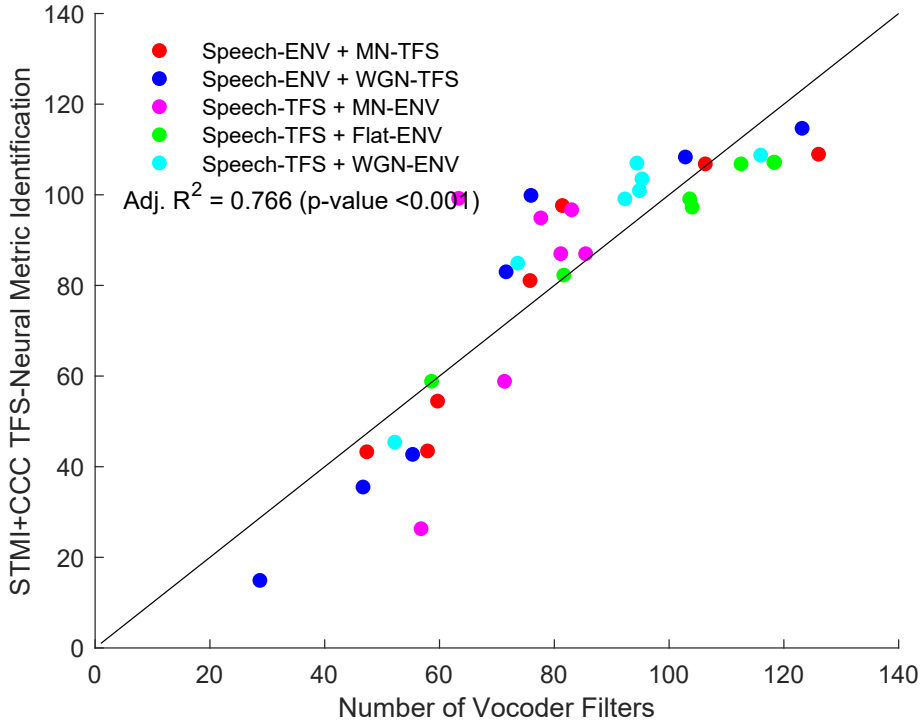

Figure 3.25: Fitting results of the predicted identification values obtained by STMI combined with CCC TFS (no interactions) and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

Observing the comparison of the predicted values and the perceptual measurements in Figure 3.25, similar to Figure 3.23, the data points in the figure are almost around the diagonal line, and the connecting lines of the data points are similar to the diagonal line, and the presence or absence of interactions does not have much effect, which indicates that the prediction results of the combination of STMI and CCC TFS are more in line with the perceptual measures, and that the algorithm is suitable for predicting the intelligibility of speech chimaeras.

In summary, by evaluating the prediction results of different algorithms, this thesis finds that applying a particular algorithm individually may be more concise, but has limited accuracy in predicting speech intelligibility. After attempting to combine the predictions of different algorithms, some combinations showed a significant improvement in prediction accuracy. Comparing the adjusted R^2 value of the different combinations, the combination of STMI with FT NSIM and the combination of STMI with CCC TFS have the highest prediction accuracy, with or without interaction, the accuracy of predicting speech intelligibility reaches 75%-76%, which means that the combination of STMI with FT NSIM and the combination of STMI with CCC TFS are both suitable to be applied to the speech chimaeras discussed in this thesis.

The STMI algorithm eliminates phase-locked features and generates chimaeras that preserve the amplitude spectrum. This makes STMI more capable of capturing ENV information. When STMI is combined with FT NSIM or CCC TFS, FT NSIM or CCC TFS can provide TFS information of the speech signals more effectively. Therefore, by accurately applying different information of the speech signals in this thesis, the accuracy of these two combinations is significantly increased.

More figures of the fitted prediction results for the various combinations of cases mentioned in this section, as well as their comparison with the perceptual measurements, are placed in Appendix A.

Chapter 4

Conclusions and Future Work

4.1 Conclusions

The goal of this thesis is to evaluate the accuracy of different algorithms in predicting speech intelligibility of speech chimaeras. To achieve this goal, this thesis studies the decomposition of speech signals into ENV and TFS, and the combination of these two parts in different signals to obtain the research object, speech chimaeras. Building on the speech recognition experiments of Wirtzfeld et al. (2017), it then investigates how mean-rate and spike-timing cues contribute to speech perception in general, and under what circumstances the number of vocoders can significantly affect the prediction accuracy. It is shown that the number of vocoders in a speech TFS chimaeras does not significantly affect the TFS spike-timing response.

In conjunction with the neurograms generated by speech chimaeras, this thesis uses STMI, NSIM and Cross-Correlation Coefficients to predict speech intelligibility for speech chimaeras, respectively. By calculating the regression models for different combinations, the combination of STMI with FT NSIM and the combination of STMI

with CCC TFS perform better than the other combinations, and can better predict the intelligibility of speech chimaeras. The above conclusions are due to the fact that STMI is more capable of obtaining ENV features, while FT NSIM and CCC TFS are all good at obtaining TFS features. By fully combining the ENV and TFS information of the speech signal, this thesis can predict speech intelligibility better, and the prediction results obtained by two good combinations are very similar.

4.2 Future Work

This thesis focuses on predicting the intelligibility of speech chimaeras under quiet conditions, but absolutely quiet conditions are not common in reality, so future work will first consider how to still predict speech intelligibility well under noisy conditions. Meanwhile, how the contribution of ENV and TFS in the prediction changes when the background environment changes, and whether the effect of the number of vocoders remains the same, these are also the research points for future work.

In the study of this thesis, there is a significant difference between WGN and MN used in speech chimaeras in prediction, and the MN ENV will be more likely to be underestimated compared to the overall flattening of Flat ENV and WGN ENV chimaeras, so in future work, it is possible to consider the choice of other noise types or how to use MN more reasonably, in order to achieve a more desirable effect.

Appendix A

Fitted Combination Figures

Table A.1: Linear regression models with the combination of MR NSIM and CCC predictions

MR NSIM and CCC ENV		**MR NSIM and CCC TFS**	
Adj.R^2	0.479	Adj.R^2	0.470
p value	<0.001	p value	<0.001

Table A.2: Linear regression models with the combination of FT NSIM and CCC predictions

FT NSIM and CCC ENV		**FT NSIM and CCC TFS**	
Adj.R^2	0.496	Adj.R^2	0.345
p value	<0.001	p value	<0.001

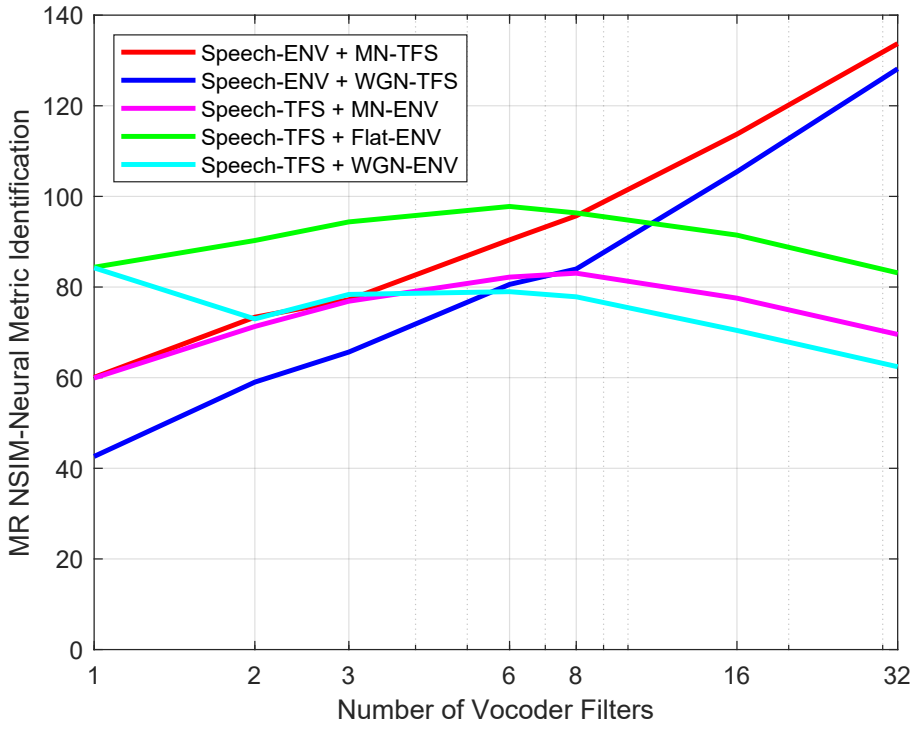

Figure A.1: MR NSIM Predictions obtained after fitting as a function of the number of vocoder filters.

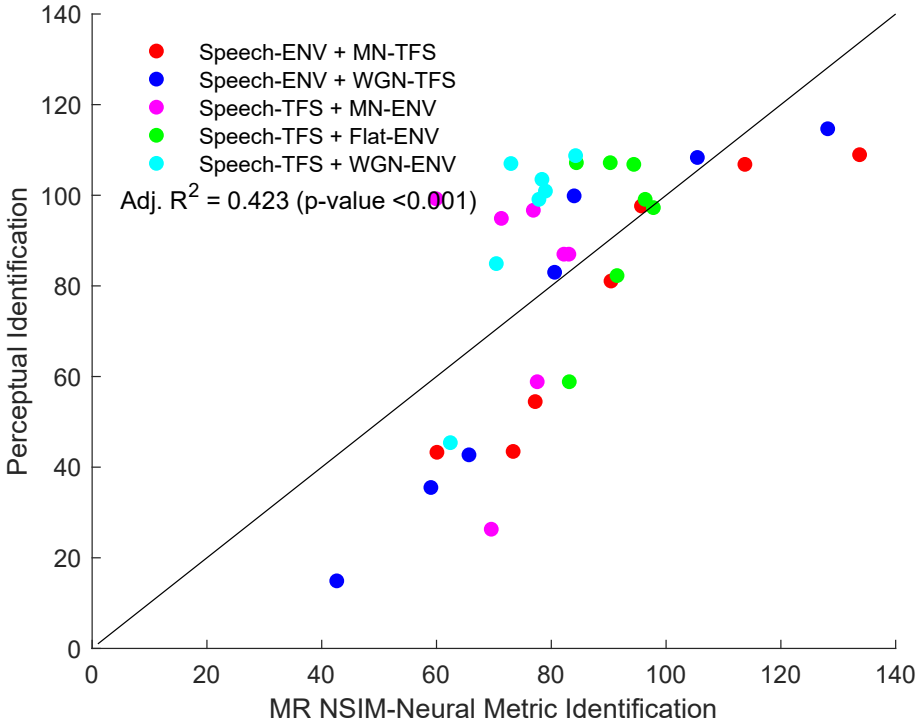

Figure A.2: Fitting results of the predicted identification values obtained by MR NSIM and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

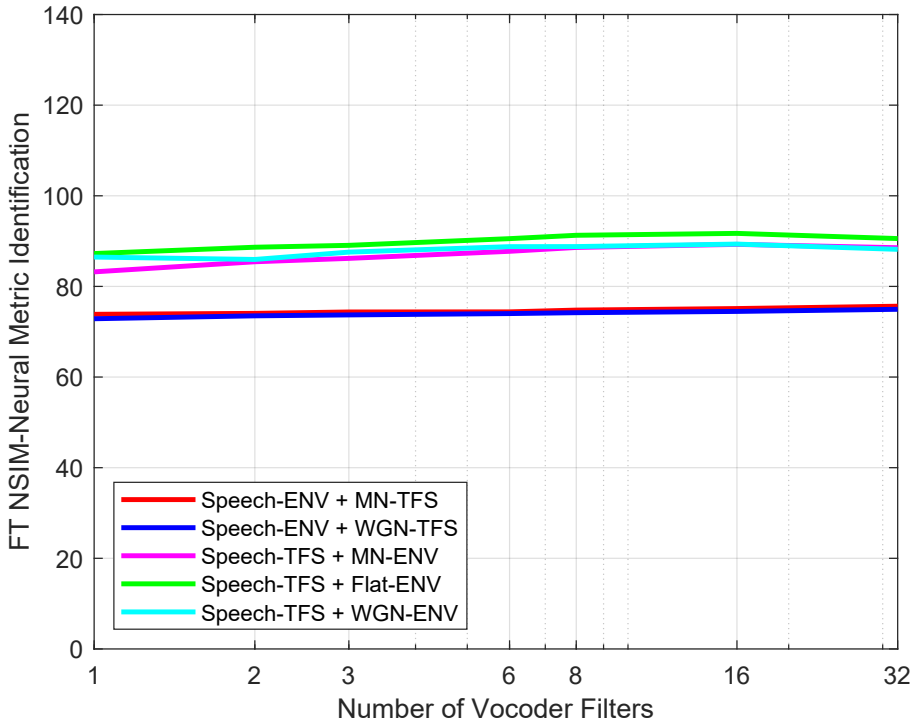

Figure A.3: FT NSIM Predictions obtained after fitting as a function of the number of vocoder filters.

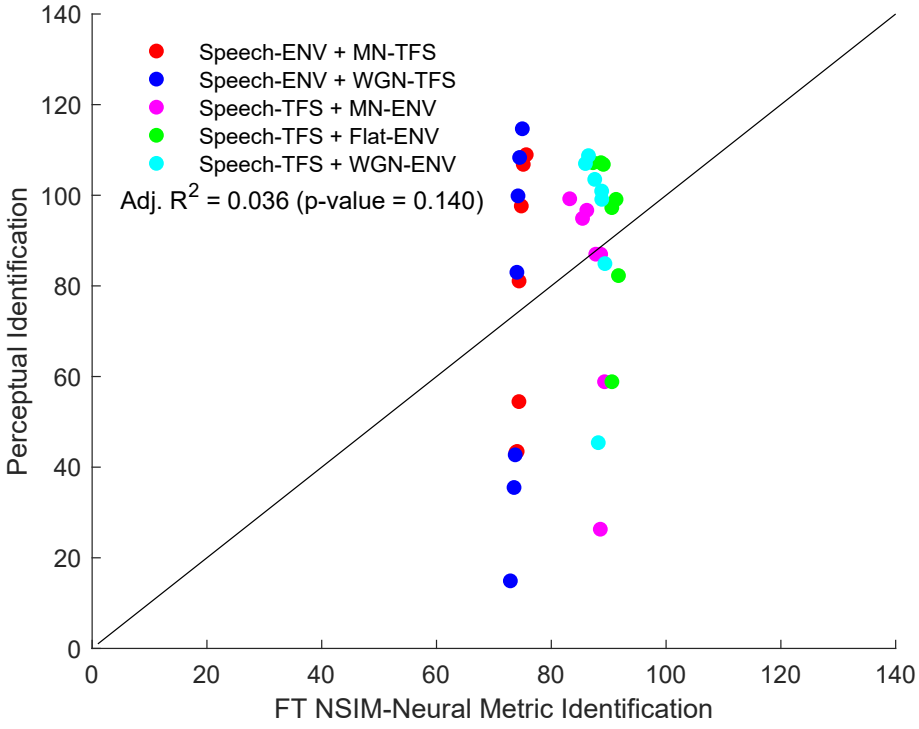

Figure A.4: Fitting results of the predicted identification values obtained by FT NSIM and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

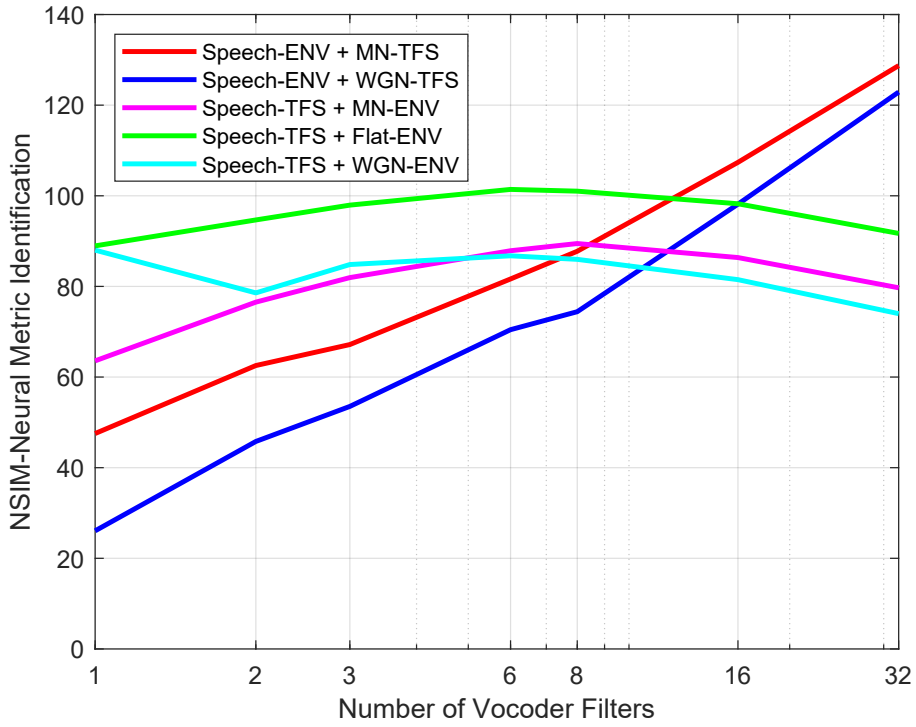

Figure A.5: MR NSIM combined with FT NSIM Predictions obtained after fitting ignoring the interaction case as a function of the number of vocoder filters.

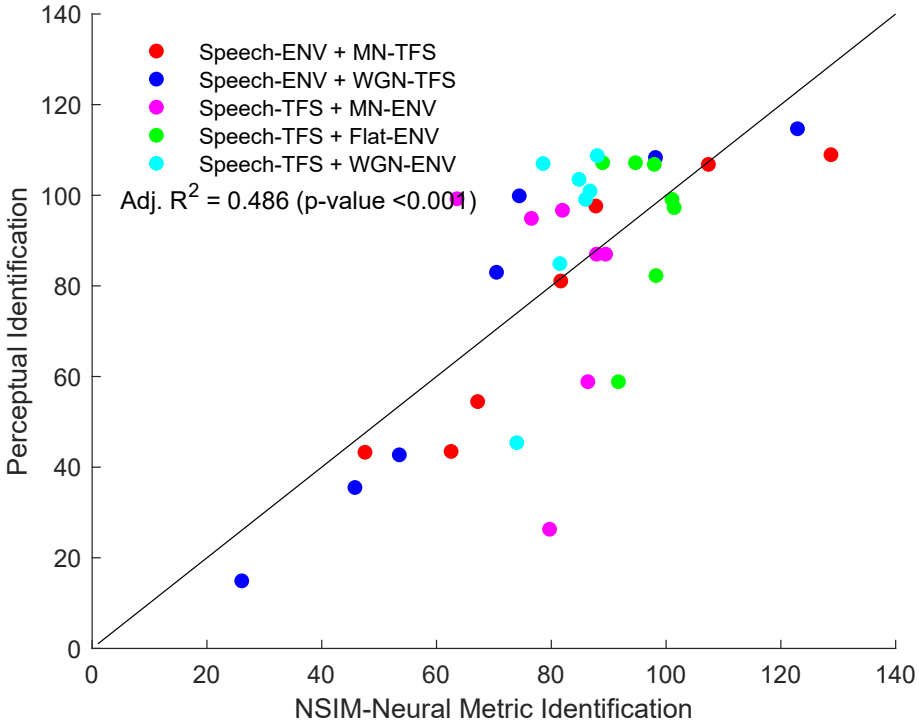

Figure A.6: Fitting results of the predicted identification values obtained by MR NSIM combined with FT NSIM and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

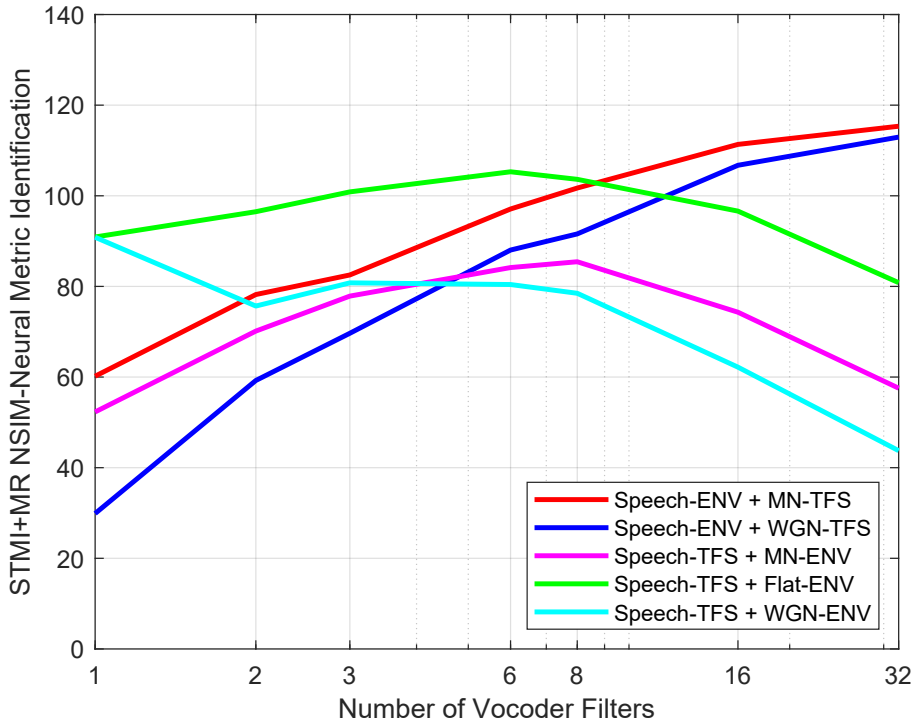

Figure A.7: STMI combined with MR NSIM Predictions obtained after fitting ignoring the interaction case as a function of the number of vocoder filters.

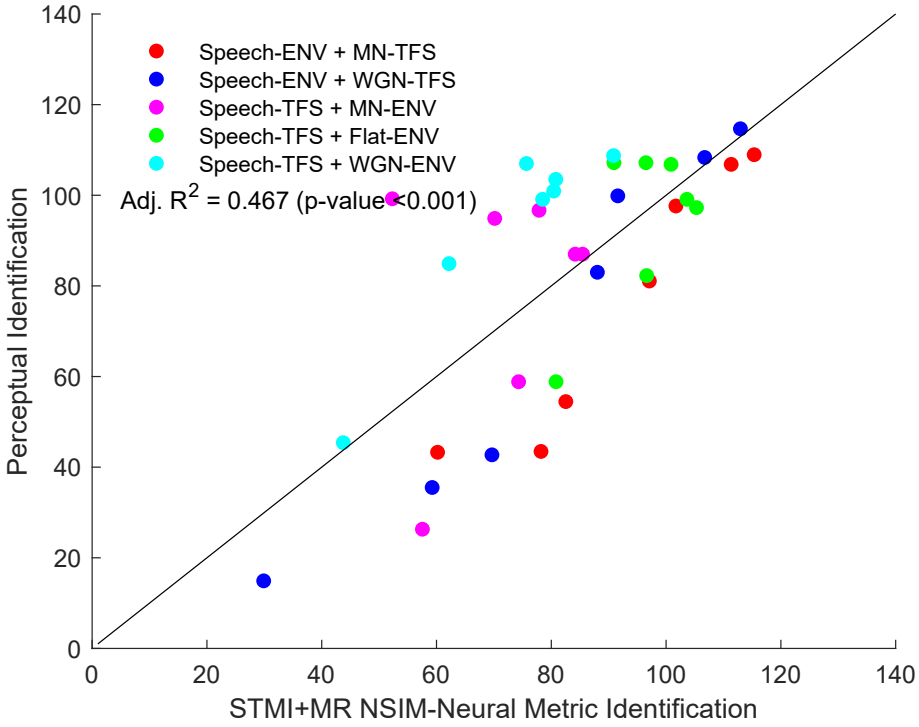

Figure A.8: Fitting results of the predicted identification values obtained by STMI combined with MR NSIM and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

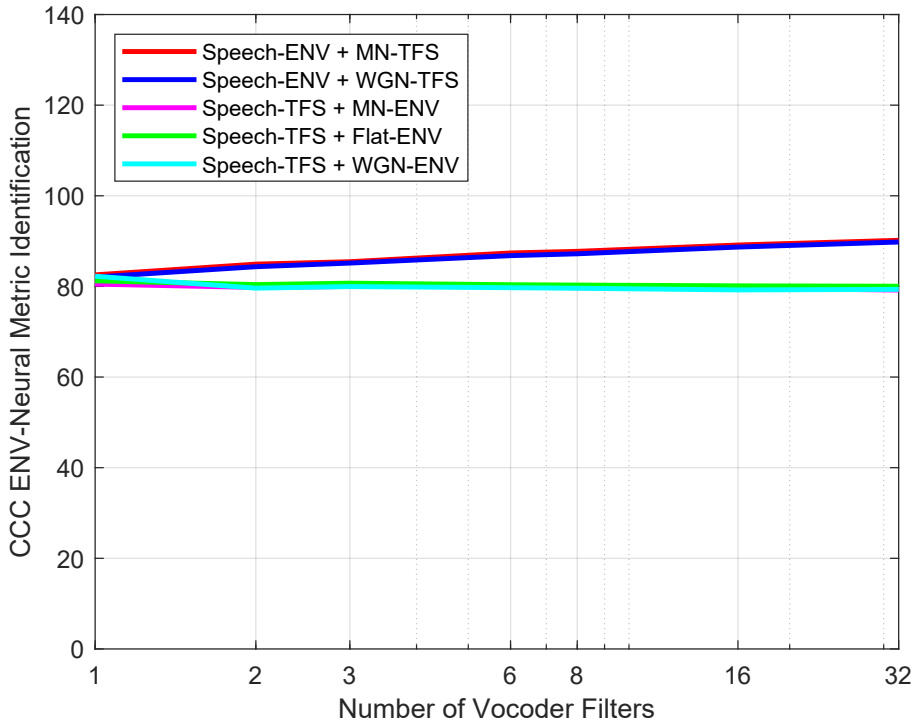

Figure A.9: CCC ENV Predictions obtained after fitting as a function of the number of vocoder filters.

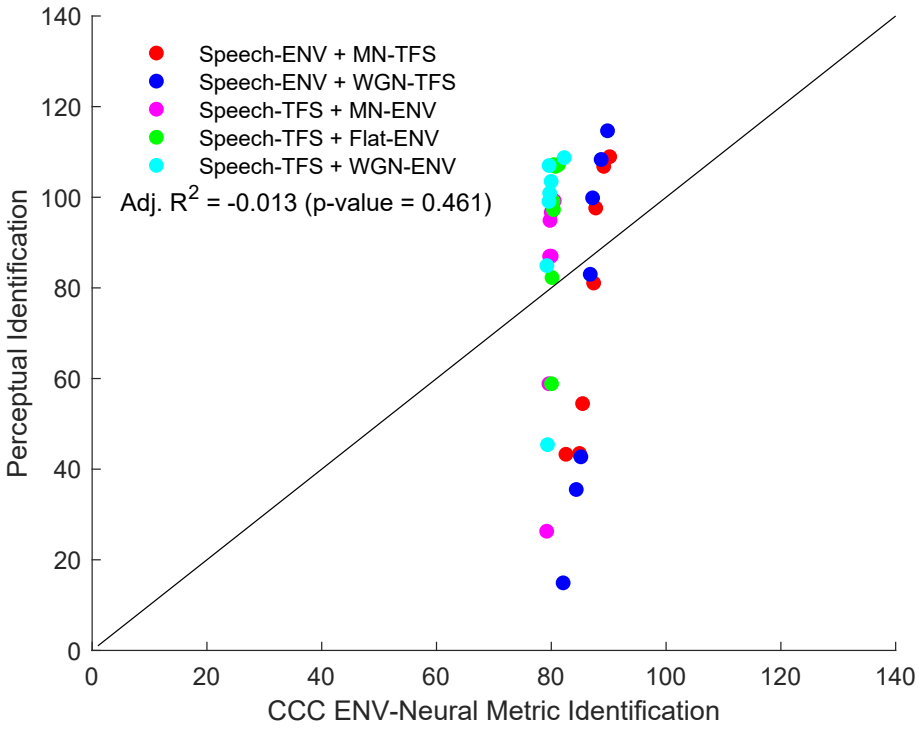

Figure A.10: Fitting results of the predicted identification values obtained by CCC ENV and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

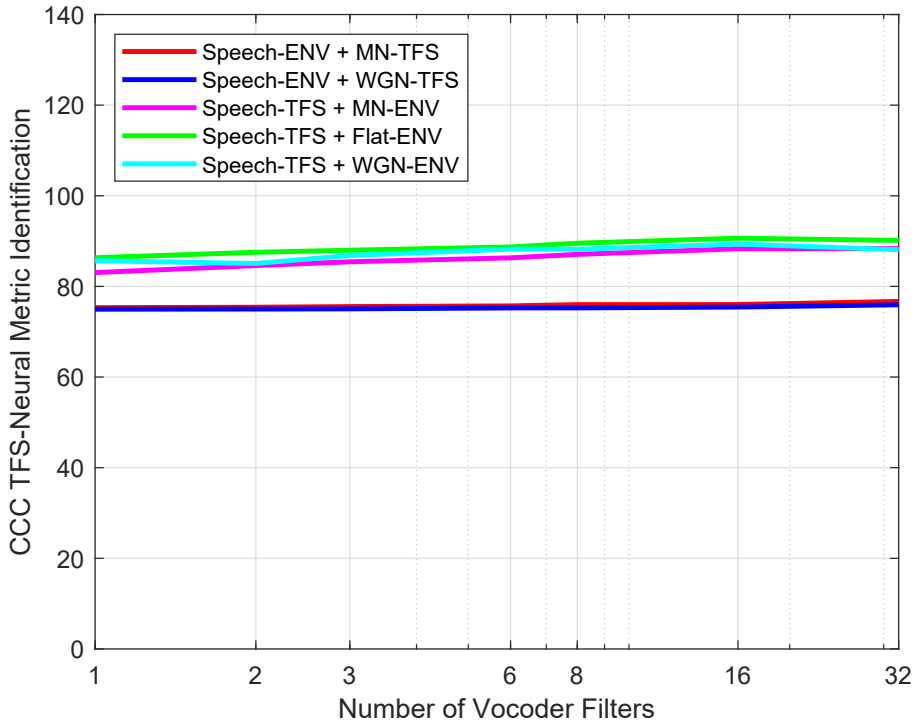

Figure A.11: CCC TFS Predictions obtained after fitting as a function of the number of vocoder filters.

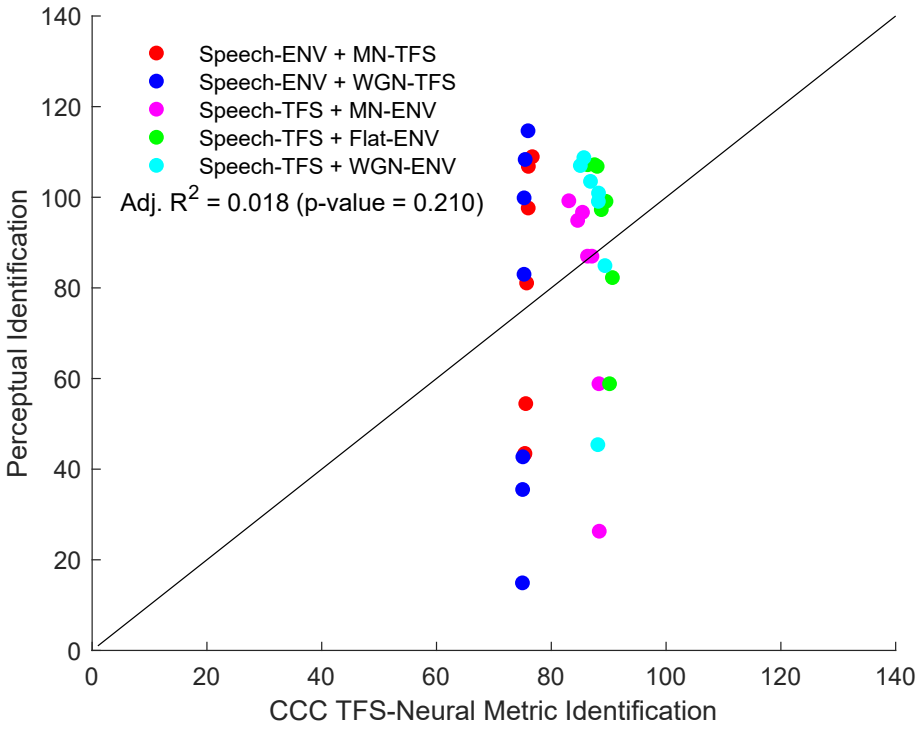

Figure A.12: Fitting results of the predicted identification values obtained by CCC TFS and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

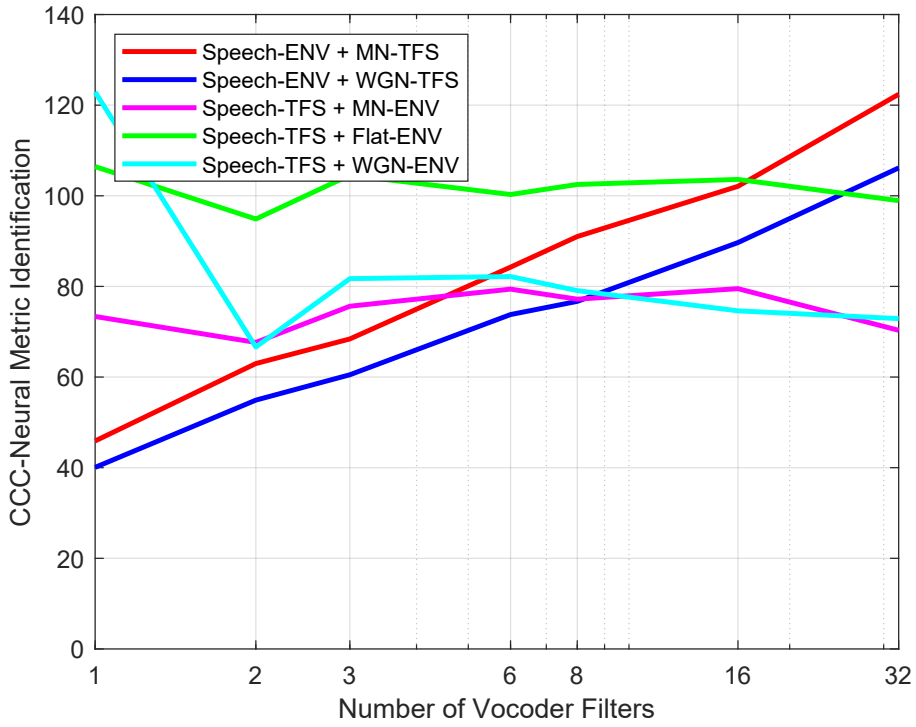

Figure A.13: CCC ENV combined with CCC TFS Predictions obtained after fitting ignoring the interaction case as a function of the number of vocoder filters.

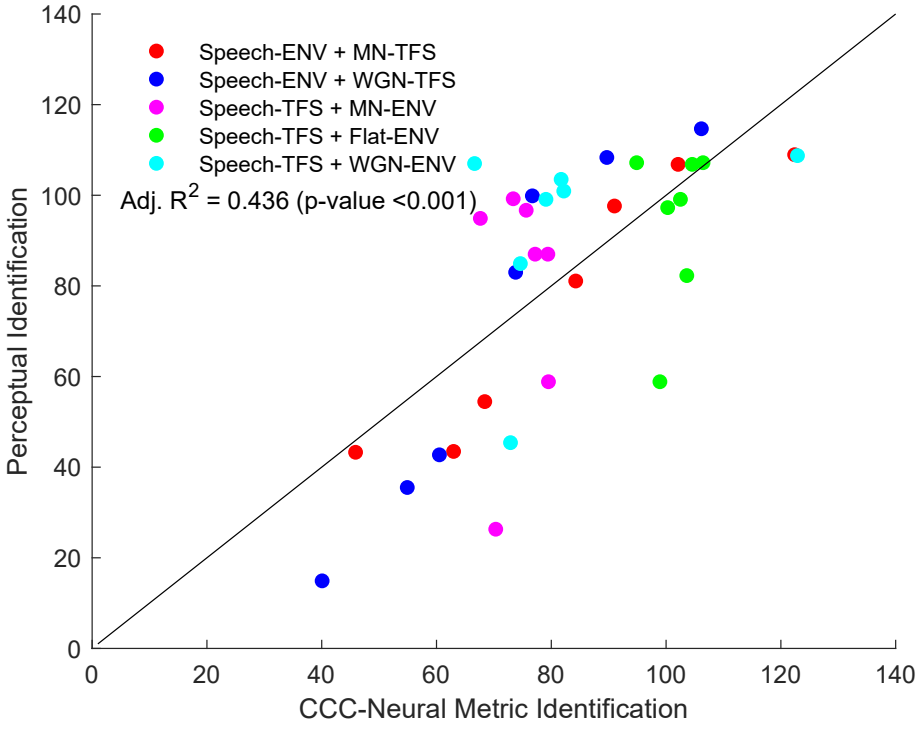

Figure A.14: Fitting results of the predicted identification values obtained by CCC ENV combined with CCC TFS and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

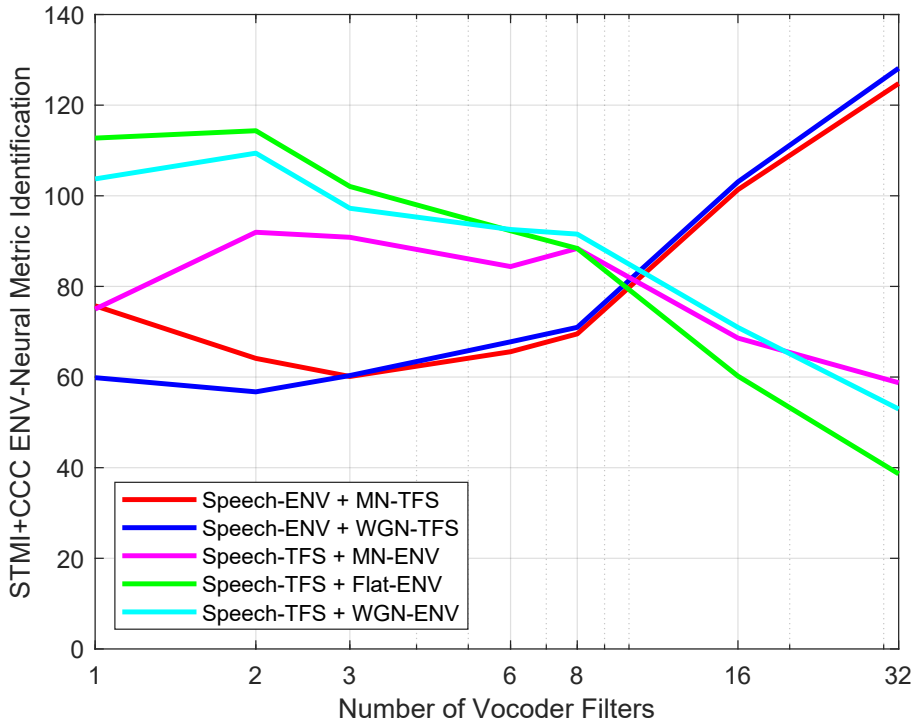

Figure A.15: STMI combined with CCC ENV Predictions obtained after fitting ignoring the interaction case as a function of the number of vocoder filters.

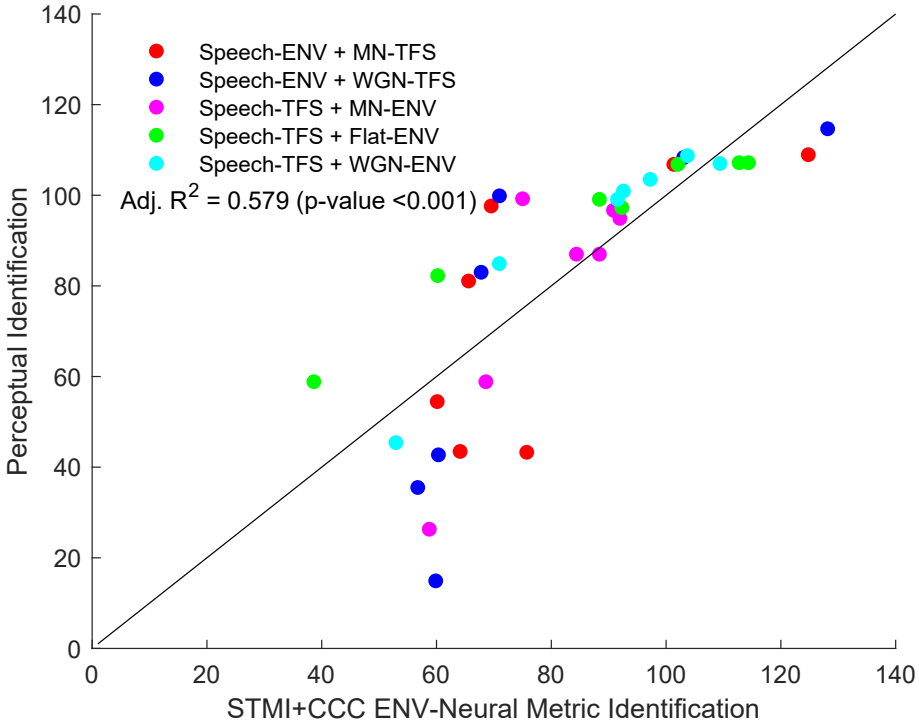

Figure A.16: Fitting results of the predicted identification values obtained by STMI combined with CCC ENV and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

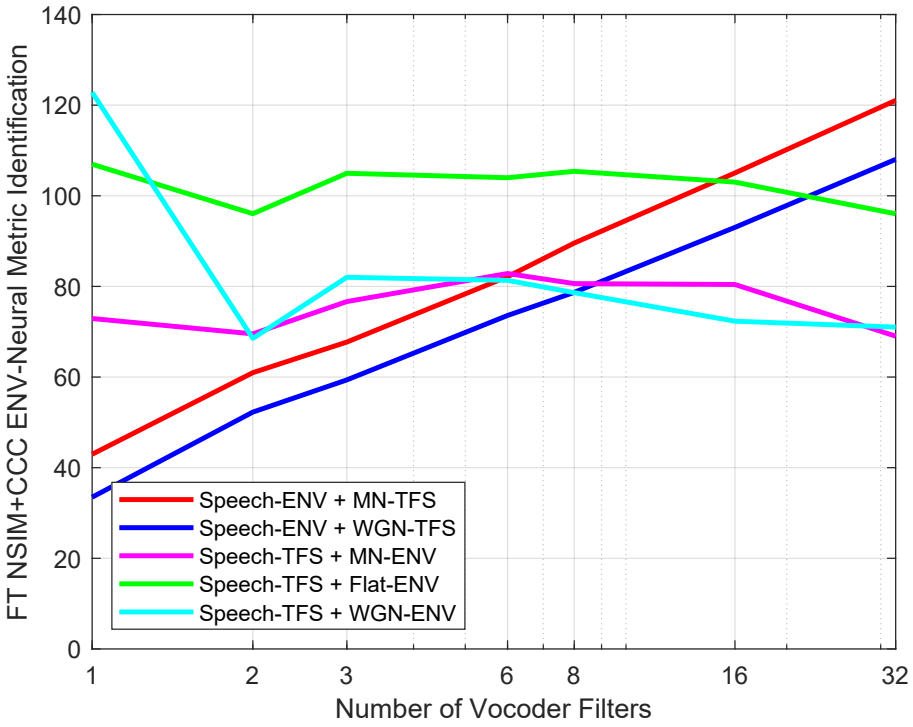

Figure A.17: FT NSIM combined with CCC ENV Predictions obtained after fitting ignoring the interaction case as a function of the number of vocoder filters.

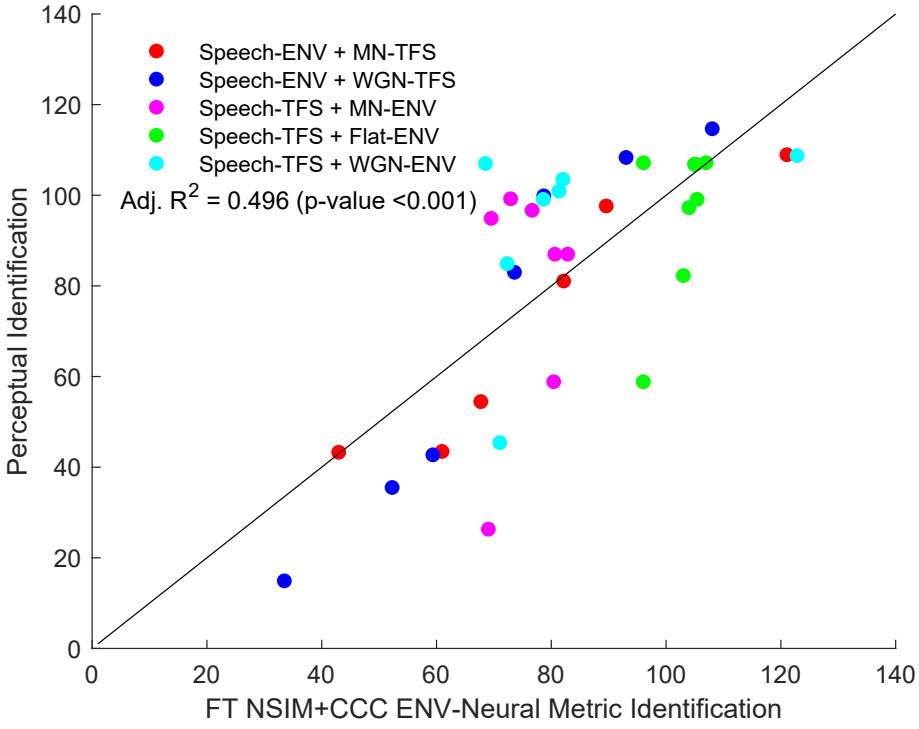

Figure A.18: Fitting results of the predicted identification values obtained by FT NSIM combined with CCC ENV and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

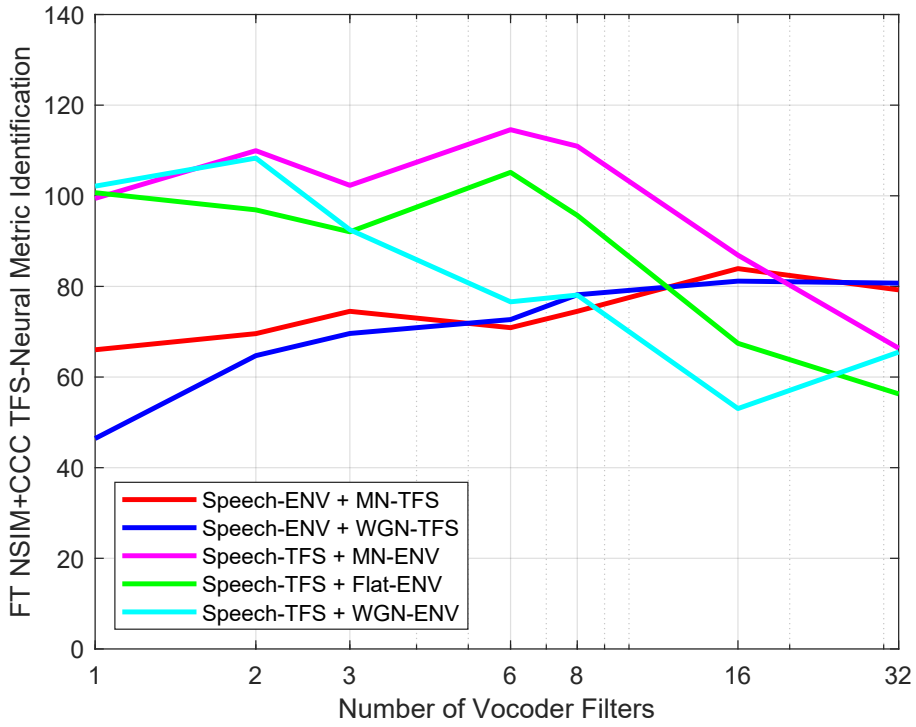

Figure A.19: FT NSIM combined with CCC TFS Predictions obtained after fitting ignoring the interaction case as a function of the number of vocoder filters.

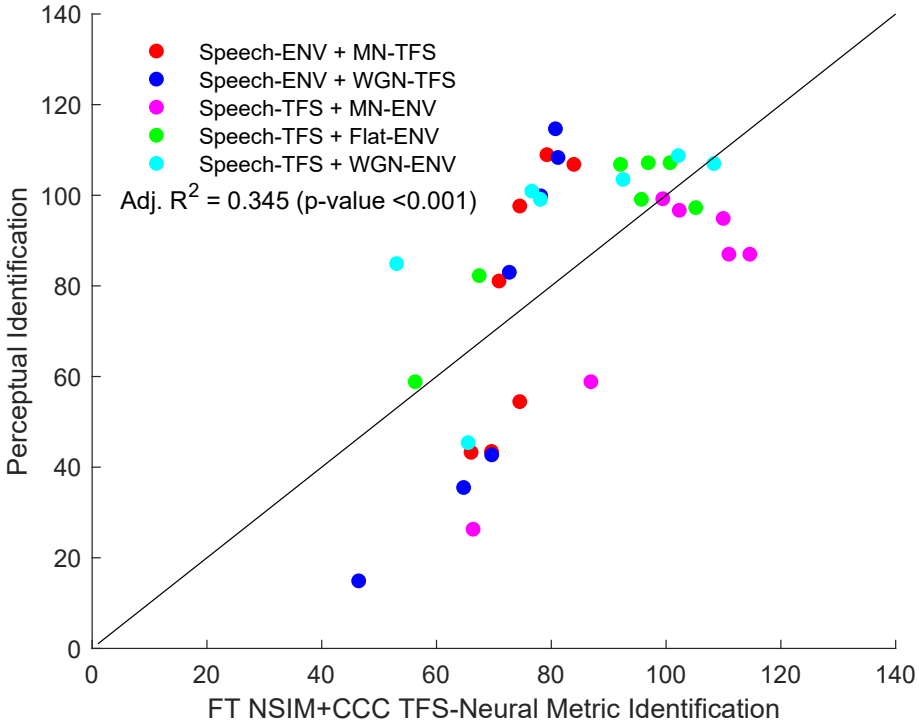

Figure A.20: Fitting results of the predicted identification values obtained by FT NSIM combined with CCC TFS and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

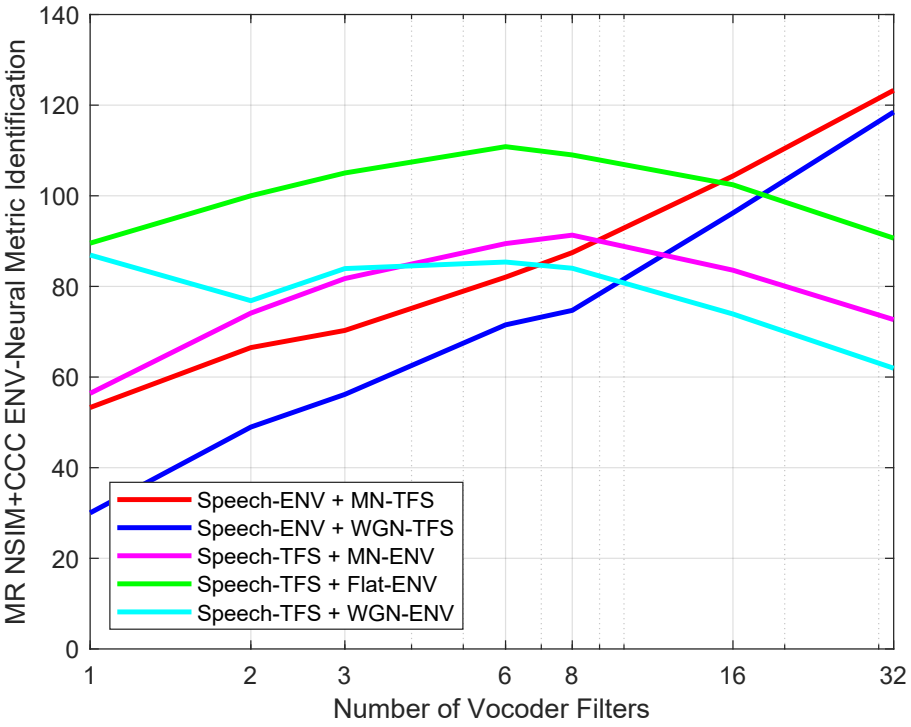

Figure A.21: MR NSIM combined with CCC ENV Predictions obtained after fitting ignoring the interaction case as a function of the number of vocoder filters.

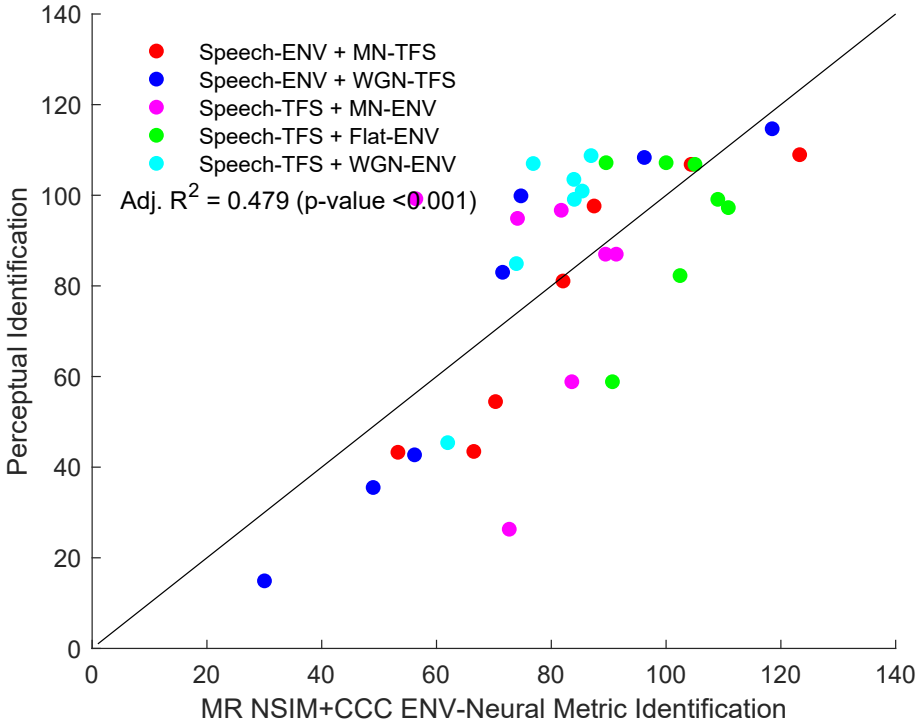

Figure A.22: Fitting results of the predicted identification values obtained by MR NSIM combined with CCC ENV and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

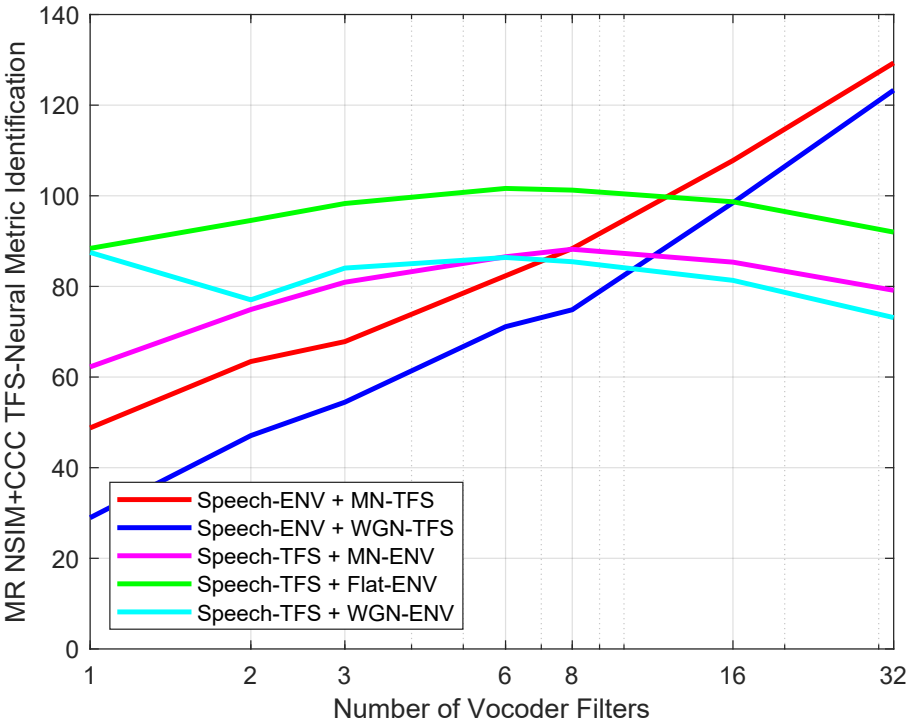

Figure A.23: MR NSIM combined with CCC TFS Predictions obtained after fitting ignoring the interaction case as a function of the number of vocoder filters.

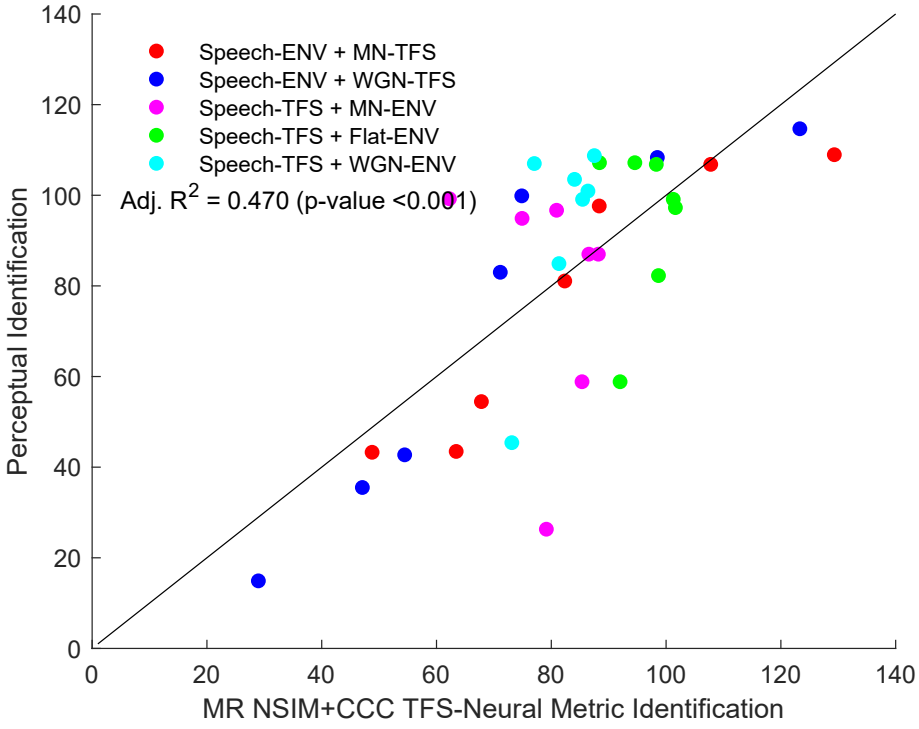

Figure A.24: Fitting results of the predicted identification values obtained by MR NSIM combined with CCC TFS and the perceptual measured identification values. The adjusted R^2 value and p value for each regression are shown in the upper lefthand corner of its respective panel.

Appendix B

Codes

B.1 STMI code

```
% Check to see if running under Matlab or Octave
if exist ('OCTAVE_VERSION', 'builtin') ~= 0
  pkg load signal;
  if exist('rms')<1
    rms = @(x) sqrt(mean(x.^2));
  end
end

if exist('parfor','builtin') % check if the Matlab Parallel
    Computation Toolbox is installed and use appropriate function
     generate_neurogram_function =
    @generate_neurogram_BEZ2018_stmi_parallelized;
     disp('Using parallelized version of neurogram generation
    function')
else
     generate_neurogram_function = @generate_neurogram_BEZ2018_stmi;
```

```
14      disp('Using serial version of neurogram generation function')
15  end
16  % Set "reference" audiogram for STMI
17  ag_fs_ref = [0 20e3];
18  ag_dbloss_ref = [0 0]; % Normal hearing
19
20  % Set "test" audiogram for STMI
21  ag_fs_test = [0 20e3];
22  ag_dbloss_test = [0 0]; % Normal hearing
23
24  species = 2; % human model with tuning based on Shera et al. (2002)
25
26  rv = 2.^(1:0.5:5);  % cortical (temporal) modulation filter rates
27  sv = 2.^(-2:0.5:3); % cortical (spectral) modulation filter scales
28
29  stimdb = 65; % speech level in dB SPL
30  SNR = 0; % in dB
31  % SNR = inf; % in dB; inf -> no background noise
32
33  load('FS.mat')
34  filter_n = [1 2 3 6 8 16 32];
35
36  numoffilter = length(filter_n);
37  STMI_1=zeros(7,50);
38  STMI_2=zeros(7,50);
39  STMI_3=zeros(7,50);
40  STMI_4=zeros(7,50);
41  STMI_5=zeros(7,50);
42
```

```
% Speech-ENV + MN-TFS
for lp = 1:numoffilter
    for i=1:50
        d=dir(['E_matched_noise_TFS' num2str(filter_n(lp)) 'filters'
    '*.mat'])
        load([d(i).name])
        str=d(i).name;
        ori_str=extractAfter(str,"filters_")
        load([ori_str])
        teststim=added_filter_bands';
        refstim=x;
        Fs_stim=FS;
        refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
        teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);

        [neurogram_stmi_ref,t_stmi,CFs] =
    generate_neurogram_function(refstim,Fs_stim,species,ag_fs_ref,
    ag_dbloss_ref);
        base_sig_ref = generate_base_signal(refstim);
        [neurogram_base_ref,t_stmi,CFs] =
    generate_neurogram_function(base_sig_ref,Fs_stim,species,
    ag_fs_ref,ag_dbloss_ref);

        [neurogram_stmi_test,t_stmi,CFs] =
    generate_neurogram_function(teststim,Fs_stim,species,ag_fs_test,
    ag_dbloss_test);
        base_sig_test = generate_base_signal(teststim);
```

```
63      [neurogram_base_test,t_stmi,CFs] =
    generate_neurogram_function(base_sig_test,Fs_stim,species,
    ag_fs_test,ag_dbloss_test);
64
65      response_endtime = length(refstim)/Fs_stim+10e-3; %
    Calculate end time of the neural response to the stimulus
66      [tmp, ind_stmi] = min(abs(t_stmi-response_endtime)); % Find
    the corresponding time index in the stmi neurogram
67
68      % Generate the cortical responses to the reference signal
    and reference base signal neurograms
69      % compute the reference "template" T matrix
70      a1_stmi_ref = abs(ngram2cortex(neurogram_stmi_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
71      a1_base_ref = abs(ngram2cortex(neurogram_base_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
72      T  = max(a1_stmi_ref - a1_base_ref,0);
73      TT = sum(sum(T(:).^2));
74
75      % Generate the cortical responses to the test signal and
    test base signal neurograms
76      % compute the test "noisy" N matrix
```

```
77          a1_stmi_test = abs(ngram2cortex(neurogram_stmi_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
78          a1_base_test = abs(ngram2cortex(neurogram_base_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
79          N   = max(a1_stmi_test - a1_base_test,0);
80          NN = sum(sum((max(T(:)-N(:),0)).^2));
81
82          STMI_1(lp,i) = 1-NN/TT
83      end
84  end
85  stmi1mean=mean(STMI_1,2);
86  stmi1std=std(STMI_1,1,2);
87
88  % Speech-ENV + WGN-TFS
89  for lp = 1:numoffilter
90      for i=1:50
91          d=dir(['E_WGN_TFS_' num2str(filter_n(lp)) 'filter' '*.mat'])
92          load([d(i).name])
93          str=d(i).name;
94          ori_str=extractAfter(str,"filter_")
95          load([ori_str])
96          teststim=added_filter_bands';
97          refstim=x;
98          Fs_stim=FS;
99          refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
```

```
100         teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);
101
102         [neurogram_stmi_ref,t_stmi,CFs] =
    generate_neurogram_function(refstim,Fs_stim,species,ag_fs_ref,
    ag_dbloss_ref);
103         base_sig_ref = generate_base_signal(refstim);
104         [neurogram_base_ref,t_stmi,CFs] =
    generate_neurogram_function(base_sig_ref,Fs_stim,species,
    ag_fs_ref,ag_dbloss_ref);
105
106         [neurogram_stmi_test,t_stmi,CFs] =
    generate_neurogram_function(teststim,Fs_stim,species,ag_fs_test,
    ag_dbloss_test);
107         base_sig_test = generate_base_signal(teststim);
108         [neurogram_base_test,t_stmi,CFs] =
    generate_neurogram_function(base_sig_test,Fs_stim,species,
    ag_fs_test,ag_dbloss_test);
109
110         response_endtime = length(refstim)/Fs_stim+10e-3; %
    Calculate end time of the neural response to the stimulus
111         [tmp, ind_stmi] = min(abs(t_stmi-response_endtime)); % Find
    the corresponding time index in the stmi neurogram
112
113         % Generate the cortical responses to the reference signal
    and reference base signal neurograms and
114         % compute the reference "template" T matrix
```

```
115         a1_stmi_ref = abs(ngram2cortex(neurogram_stmi_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
116         a1_base_ref = abs(ngram2cortex(neurogram_base_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
117         T   = max(a1_stmi_ref - a1_base_ref,0);
118         TT = sum(sum(T(:).^2));
119
120         % Generate the cortical responses to the test signal and
    test base signal neurograms and
121         % compute the test "noisy" N matrix
122         a1_stmi_test = abs(ngram2cortex(neurogram_stmi_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
123         a1_base_test = abs(ngram2cortex(neurogram_base_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
124         N   = max(a1_stmi_test - a1_base_test,0);
125         NN = sum(sum((max(T(:)-N(:),0)).^2));
126
127         STMI_2(lp,i) = 1-NN/TT
128      end
129 end
130 stmi2mean=mean(STMI_2,2);
```

```
131 stmi2std=std(STMI_2,1,2);
132
133 % Speech-TFS + MN-ENV
134 for lp = 1:numoffilter
135     for i=1:50
136         d=dir(['TFS_matched_noise_E' num2str(filter_n(lp)) 'filters'
      '*.mat'])
137         load([d(i).name])
138         str=d(i).name;
139         ori_str=extractAfter(str,"filters_")
140         load([ori_str])
141         teststim=added_filter_bands';
142         refstim=x;
143         Fs_stim=FS;
144         refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
145         teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);
146
147         [neurogram_stmi_ref,t_stmi,CFs] =
      generate_neurogram_function(refstim,Fs_stim,species,ag_fs_ref,
      ag_dbloss_ref);
148         base_sig_ref = generate_base_signal(refstim);
149         [neurogram_base_ref,t_stmi,CFs] =
      generate_neurogram_function(base_sig_ref,Fs_stim,species,
      ag_fs_ref,ag_dbloss_ref);
150
151         [neurogram_stmi_test,t_stmi,CFs] =
      generate_neurogram_function(teststim,Fs_stim,species,ag_fs_test,
      ag_dbloss_test);
152         base_sig_test = generate_base_signal(teststim);
```

```
153         [neurogram_base_test,t_stmi,CFs] =
    generate_neurogram_function(base_sig_test,Fs_stim,species,
    ag_fs_test,ag_dbloss_test);
154
155         response_endtime = length(refstim)/Fs_stim+10e-3; %
    Calculate end time of the neural response to the stimulus
156         [tmp, ind_stmi] = min(abs(t_stmi-response_endtime)); % Find
    the corresponding time index in the stmi neurogram
157
158         % Generate the cortical responses to the reference signal
    and reference base signal neurograms and
159         % compute the reference "template" T matrix
160         a1_stmi_ref = abs(ngram2cortex(neurogram_stmi_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
161         a1_base_ref = abs(ngram2cortex(neurogram_base_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
162         T  = max(a1_stmi_ref - a1_base_ref,0);
163         TT = sum(sum(T(:).^2));
164
165         % Generate the cortical responses to the test signal and
    test base signal neurograms and
166         % compute the test "noisy" N matrix
```

```
167         a1_stmi_test = abs(ngram2cortex(neurogram_stmi_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
168         a1_base_test = abs(ngram2cortex(neurogram_base_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
169         N  = max(a1_stmi_test - a1_base_test,0);
170         NN = sum(sum((max(T(:)-N(:),0)).^2));
171
172         STMI_3(lp,i) = 1-NN/TT
173     end
174 end
175 stmi3mean=mean(STMI_3,2);
176 stmi3std=std(STMI_3,1,2);
177
178 % Speech-TFS Only
179 for lp = 1:numoffilter
180     for i=1:50
181         d=dir(['TFS_only_' num2str(filter_n(lp)) 'filters' '*.mat'])
182         load([d(i).name])
183         str=d(i).name;
184         ori_str=extractAfter(str,"filters_")
185         load([ori_str])
186         teststim=added_filter_bands';
187         refstim=x;
188         Fs_stim=FS;
189         refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
```

```
190         teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);
191
192         [neurogram_stmi_ref,t_stmi,CFs] =
    generate_neurogram_function(refstim,Fs_stim,species,ag_fs_ref,
    ag_dbloss_ref);
193         base_sig_ref = generate_base_signal(refstim);
194         [neurogram_base_ref,t_stmi,CFs] =
    generate_neurogram_function(base_sig_ref,Fs_stim,species,
    ag_fs_ref,ag_dbloss_ref);
195
196         [neurogram_stmi_test,t_stmi,CFs] =
    generate_neurogram_function(teststim,Fs_stim,species,ag_fs_test,
    ag_dbloss_test);
197         base_sig_test = generate_base_signal(teststim);
198         [neurogram_base_test,t_stmi,CFs] =
    generate_neurogram_function(base_sig_test,Fs_stim,species,
    ag_fs_test,ag_dbloss_test);
199
200         response_endtime = length(refstim)/Fs_stim+10e-3; %
    Calculate end time of the neural response to the stimulus
201         [tmp, ind_stmi] = min(abs(t_stmi-response_endtime)); % Find
    the corresponding time index in the stmi neurogram
202
203         % Generate the cortical responses to the reference signal
    and reference base signal neurograms and
204         % compute the reference "template" T matrix
```

```
205         a1_stmi_ref = abs(ngram2cortex(neurogram_stmi_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
206         a1_base_ref = abs(ngram2cortex(neurogram_base_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
207         T   = max(a1_stmi_ref - a1_base_ref,0);
208         TT = sum(sum(T(:).^2));
209
210         % Generate the cortical responses to the test signal and
    test base signal neurograms and
211         % compute the test "noisy" N matrix
212         a1_stmi_test = abs(ngram2cortex(neurogram_stmi_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
213         a1_base_test = abs(ngram2cortex(neurogram_base_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
214         N   = max(a1_stmi_test - a1_base_test,0);
215         NN = sum(sum((max(T(:)-N(:),0)).^2));
216
217         STMI_4(lp,i) = 1-NN/TT
218      end
219 end
220 stmi4mean=mean(STMI_4,2);
```

```
221 stmi4std=std(STMI_4,1,2);
222
223 % Speech-TFS + WGN-ENV
224 for lp = 1:numoffilter
225     for i=1:50
226         d=dir(['TFS_WGN_E_' num2str(filter_n(lp)) 'filter' '*.mat'])
227         load([d(i).name])
228         str=d(i).name;
229         ori_str=extractAfter(str,"filter_")
230         load([ori_str])
231         teststim=added_filter_bands';
232         refstim=x;
233         Fs_stim=FS;
234         refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
235         teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);
236
237         [neurogram_stmi_ref,t_stmi,CFs] =
    generate_neurogram_function(refstim,Fs_stim,species,ag_fs_ref,
    ag_dbloss_ref);
238         base_sig_ref = generate_base_signal(refstim);
239         [neurogram_base_ref,t_stmi,CFs] =
    generate_neurogram_function(base_sig_ref,Fs_stim,species,
    ag_fs_ref,ag_dbloss_ref);
240
241         [neurogram_stmi_test,t_stmi,CFs] =
    generate_neurogram_function(teststim,Fs_stim,species,ag_fs_test,
    ag_dbloss_test);
242         base_sig_test = generate_base_signal(teststim);
```

```
243         [neurogram_base_test,t_stmi,CFs] =
    generate_neurogram_function(base_sig_test,Fs_stim,species,
    ag_fs_test,ag_dbloss_test);
244
245         response_endtime = length(refstim)/Fs_stim+10e-3; %
    Calculate end time of the neural response to the stimulus
246         [tmp, ind_stmi] = min(abs(t_stmi-response_endtime)); % Find
    the corresponding time index in the stmi neurogram
247
248         % Generate the cortical responses to the reference signal
    and reference base signal neurograms and
249         % compute the reference "template" T matrix
250         a1_stmi_ref = abs(ngram2cortex(neurogram_stmi_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
251         a1_base_ref = abs(ngram2cortex(neurogram_base_ref(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
252         T  = max(a1_stmi_ref - a1_base_ref,0);
253         TT = sum(sum(T(:).^2));
254
255         % Generate the cortical responses to the test signal and
    test base signal neurograms and
256         % compute the test "noisy" N matrix
```

```
257         a1_stmi_test = abs(ngram2cortex(neurogram_stmi_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
258         a1_base_test = abs(ngram2cortex(neurogram_base_test(:,1:
    ind_stmi)',diff(t_stmi(1:2)),rv,sv)); % Neurogram needs to be
    transposed so the response for each CF is in a column -> 128
    columns
259         N  = max(a1_stmi_test - a1_base_test,0);
260         NN = sum(sum((max(T(:)-N(:),0)).^2));
261
262         STMI_5(lp,i) = 1-NN/TT
263     end
264 end
265 stmi5mean=mean(STMI_5,2);
266 stmi5std=std(STMI_5,1,2);
267
268 figure
269
270 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
271    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
272    'Speech-TFS + WGN-ENV'};
273
274 errorbar(filter_n,stmi1mean,stmi1std/sqrt(50),'r','linewidth',2.0)
275 hold on
276 errorbar(filter_n,stmi2mean,stmi2std/sqrt(50),'b','linewidth',2.0)
277 hold on
278 errorbar(filter_n,stmi3mean,stmi3std/sqrt(50),'m','linewidth',2.0)
279 hold on
```

```
errorbar(filter_n,stmi4mean,stmi4std/sqrt(50),'g','linewidth',2.0)
hold on
errorbar(filter_n,stmi5mean,stmi5std/sqrt(50),'c','linewidth',2.0)
ylim([0 1])
xlabel('Number of Vocoder Filters')
ylabel('STMI')
xlim([0.9 34])
set(gca,'XTick',filter_n)
set(gca,'xscale','log')
grid on
legend(casestrs([1 2 3 4 5]),'location','SouthWest')
```

B.2 NSIM code

```
% Check to see if running under Matlab or Octave
if exist ('OCTAVE_VERSION', 'builtin') ~= 0
  pkg load signal;
  if exist('rms')<1
    rms = @(x) sqrt(mean(x.^2));
  end
end

if exist('parfor','builtin') % check if the Matlab Parallel
    Computation
                             % Toolbox is installed and use
    appropriate
                             % function
     generate_neurogram_function =
    @generate_neurogram_BEZ2018_parallelized;
```

```
13      disp('Using parallelized version of neurogram generation
        function')
14  else
15      generate_neurogram_function = @generate_neurogram_BEZ2018;
16      disp('Using serial version of neurogram generation function')
17  end
18
19  % Set "reference" audiogram
20  ag_fs_ref = [0 20e3];
21  ag_dbloss_ref = [0 0]; % Normal hearing
22
23  % Set "test" audiogram
24  ag_fs_test = [125 250 500 1e3 2e3 4e3 8e3];
25  ag_dbloss_test = [0 0 0 0 0 0 0]; % Normal hearing
26
27  species = 2;
28
29  % NSIM parameters
30  weights.alpha = 1.0;
31  weights.beta = 0.0;
32  weights.gamma = 1.0;
33  window_type = 0;
34  win3x3 = ones(3,3);
35
36  stimdb = 65; % speech level in dB SPL
37  SNR = 0; % in dB
38  % SNR = inf; % in dB; inf -> no background noise
39
40  load('FS.mat')
```

```
41 filter_n = [1 2 3 6 8 16 32];
42
43 numoffilter = length(filter_n);
44 NSIM_MR_1=zeros(7,50);
45 NSIM_MR_2=zeros(7,50);
46 NSIM_MR_3=zeros(7,50);
47 NSIM_MR_4=zeros(7,50);
48 NSIM_MR_5=zeros(7,50);
49 NSIM_FT_1=zeros(7,50);
50 NSIM_FT_2=zeros(7,50);
51 NSIM_FT_3=zeros(7,50);
52 NSIM_FT_4=zeros(7,50);
53 NSIM_FT_5=zeros(7,50);
54
55 % Speech-ENV + MN-TFS
56 for lp = 1:numoffilter
57     for i=1:50
58         d=dir(['E_matched_noise_TFS' num2str(filter_n(lp)) 'filters'
     '*.mat'])
59         load([d(i).name])
60         str=d(i).name;
61         ori_str=extractAfter(str,"filters_")
62         load([ori_str])
63         teststim=added_filter_bands';
64         refstim=x;
65         Fs_stim=FS;
66         refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
67         teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);
68
```

```
69      [neurogram_ft_ref,neurogram_mr_ref,neurogram_Sout_ref
,~,~,~,~] = generate_neurogram_function(refstim,Fs_stim,species,
ag_fs_ref,ag_dbloss_ref);
70      [neurogram_ft_test,neurogram_mr_test,neurogram_Sout_test,
t_ft,t_mr,t_Sout,CFs] = generate_neurogram_function(teststim,
Fs_stim,species,ag_fs_test,ag_dbloss_test);
71
72      response_endtime = length(refstim)/Fs_stim+10e-3; %
Calculate end time of the neural response to the stimulus
73      [tmp, ind_mr] = min(abs(t_mr-response_endtime)); % Find the
corresponding time index in the mr neurogram
74      [tmp, ind_ft] = min(abs(t_ft-response_endtime)); % Find the
corresponding time index in the ft neurogram
75
76      % Scaling method used by Hines and Harte (Speech Comm 2010,
2012)
77      % scl_mr = 255/max(max(neurogram_mr_ref(:,1:ind_mr)));
78      % scl_ft = 255/max(max(neurogram_ft_ref(:,1:ind_ft)));
79
80      % New scaling method developed by M. R. Wirtzfeld (see
Wirtzfeld et al., JARO 2017)
81      scl_mr = 1/50/t_mr(2);
82      scl_ft = 1/50/t_ft(2);
83
84      [NSIM_MR_1(lp,i), ssim_mr] = mssim_ians(scl_mr*
neurogram_mr_ref(:,1:ind_mr), scl_mr*neurogram_mr_test(:,1:ind_mr
), weights, win3x3, window_type );
```

```
85          [NSIM_FT_1(lp,i), ssim_ft] = mssim_ians(scl_ft*
    neurogram_ft_ref(:,1:ind_ft), scl_ft*neurogram_ft_test(:,1:ind_ft
    ), weights, win3x3, window_type );
86      end
87  end
88  nsim1mean_mr=mean(NSIM_MR_1,2);
89  nsim1std_mr=std(NSIM_MR_1,1,2);
90  nsim1mean_ft=mean(NSIM_FT_1,2);
91  nsim1std_ft=std(NSIM_FT_1,1,2);
92
93  % Speech-ENV + WGN-TFS
94  for lp = 1:numoffilter
95      for i=1:50
96          d=dir(['E_WGN_TFS_' num2str(filter_n(lp)) 'filter' '*.mat'])
97          load([d(i).name])
98          str=d(i).name;
99          ori_str=extractAfter(str,"filter_")
100         load([ori_str])
101         teststim=added_filter_bands';
102         refstim=x;
103         Fs_stim=FS;
104         refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
105         teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);
106
107         [neurogram_ft_ref,neurogram_mr_ref,neurogram_Sout_ref
    ,~,~,~,~] = generate_neurogram_function(refstim,Fs_stim,species,
    ag_fs_ref,ag_dbloss_ref);
```

```
108         [neurogram_ft_test,neurogram_mr_test,neurogram_Sout_test,
    t_ft,t_mr,t_Sout,CFs] = generate_neurogram_function(teststim,
    Fs_stim,species,ag_fs_test,ag_dbloss_test);
109
110         response_endtime = length(refstim)/Fs_stim+10e-3; %
    Calculate end time of the neural response to the stimulus
111         [tmp, ind_mr] = min(abs(t_mr-response_endtime)); % Find the
    corresponding time index in the mr neurogram
112         [tmp, ind_ft] = min(abs(t_ft-response_endtime)); % Find the
    corresponding time index in the ft neurogram
113
114         % Scaling method used by Hines and Harte (Speech Comm 2010,
    2012)
115         % scl_mr = 255/max(max(neurogram_mr_ref(:,1:ind_mr)));
116         % scl_ft = 255/max(max(neurogram_ft_ref(:,1:ind_ft)));
117
118         % New scaling method developed by M. R. Wirtzfeld (see
    Wirtzfeld et al., JARO 2017)
119         scl_mr = 1/50/t_mr(2);
120         scl_ft = 1/50/t_ft(2);
121
122         [NSIM_MR_2(lp,i), ssim_mr] = mssim_ians(scl_mr*
    neurogram_mr_ref(:,1:ind_mr), scl_mr*neurogram_mr_test(:,1:ind_mr
    ), weights, win3x3, window_type );
123         [NSIM_FT_2(lp,i), ssim_ft] = mssim_ians(scl_ft*
    neurogram_ft_ref(:,1:ind_ft), scl_ft*neurogram_ft_test(:,1:ind_ft
    ), weights, win3x3, window_type );
124      end
125 end
```

```
126 nsim2mean_mr=mean(NSIM_MR_2,2);
127 nsim2std_mr=std(NSIM_MR_2,1,2);
128 nsim2mean_ft=mean(NSIM_FT_2,2);
129 nsim2std_ft=std(NSIM_FT_2,1,2);
130
131 % Speech-TFS + MN-ENV
132 for lp = 1:numoffilter
133     for i=1:50
134         d=dir(['TFS_matched_noise_E' num2str(filter_n(lp)) 'filters'
      '*.mat'])
135         load([d(i).name])
136         str=d(i).name;
137         ori_str=extractAfter(str,"filters_")
138         load([ori_str])
139         teststim=added_filter_bands';
140         refstim=x;
141         Fs_stim=FS;
142         refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
143         teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);
144
145         [neurogram_ft_ref,neurogram_mr_ref,neurogram_Sout_ref
     ,~,~,~,~] = generate_neurogram_function(refstim,Fs_stim,species,
     ag_fs_ref,ag_dbloss_ref);
146         [neurogram_ft_test,neurogram_mr_test,neurogram_Sout_test,
     t_ft,t_mr,t_Sout,CFs] = generate_neurogram_function(teststim,
     Fs_stim,species,ag_fs_test,ag_dbloss_test);
147
148         response_endtime = length(refstim)/Fs_stim+10e-3; %
     Calculate end time of the neural response to the stimulus
```

```
149         [tmp, ind_mr] = min(abs(t_mr-response_endtime)); % Find the
    corresponding time index in the mr neurogram
150         [tmp, ind_ft] = min(abs(t_ft-response_endtime)); % Find the
    corresponding time index in the ft neurogram
151
152         % Scaling method used by Hines and Harte (Speech Comm 2010,
    2012)
153         % scl_mr = 255/max(max(neurogram_mr_ref(:,1:ind_mr)));
154         % scl_ft = 255/max(max(neurogram_ft_ref(:,1:ind_ft)));
155
156         % New scaling method developed by M. R. Wirtzfeld (see
    Wirtzfeld et al., JARO 2017)
157         scl_mr = 1/50/t_mr(2);
158         scl_ft = 1/50/t_ft(2);
159
160         [NSIM_MR_3(lp,i), ssim_mr] = mssim_ians(scl_mr*
    neurogram_mr_ref(:,1:ind_mr), scl_mr*neurogram_mr_test(:,1:ind_mr
    ), weights, win3x3, window_type );
161         [NSIM_FT_3(lp,i), ssim_ft] = mssim_ians(scl_ft*
    neurogram_ft_ref(:,1:ind_ft), scl_ft*neurogram_ft_test(:,1:ind_ft
    ), weights, win3x3, window_type );
162     end
163 end
164 nsim3mean_mr=mean(NSIM_MR_3,2);
165 nsim3std_mr=std(NSIM_MR_3,1,2);
166 nsim3mean_ft=mean(NSIM_FT_3,2);
167 nsim3std_ft=std(NSIM_FT_3,1,2);
168
169 % Speech-TFS Only
```

```
for lp = 1:numoffilter
    for i=1:50
        d=dir(['TFS_only_' num2str(filter_n(lp)) 'filters' '*.mat'])
        load([d(i).name])
        str=d(i).name;
        ori_str=extractAfter(str,"filters_")
        load([ori_str])
        teststim=added_filter_bands';
        refstim=x;
        Fs_stim=FS;
        refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
        teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);

        [neurogram_ft_ref,neurogram_mr_ref,neurogram_Sout_ref
    ,~,~,~,~] = generate_neurogram_function(refstim,Fs_stim,species,
    ag_fs_ref,ag_dbloss_ref);
        [neurogram_ft_test,neurogram_mr_test,neurogram_Sout_test,
    t_ft,t_mr,t_Sout,CFs] = generate_neurogram_function(teststim,
    Fs_stim,species,ag_fs_test,ag_dbloss_test);

        response_endtime = length(refstim)/Fs_stim+10e-3; %
    Calculate end time of the neural response to the stimulus
        [tmp, ind_mr] = min(abs(t_mr-response_endtime)); % Find the
    corresponding time index in the mr neurogram
        [tmp, ind_ft] = min(abs(t_ft-response_endtime)); % Find the
    corresponding time index in the ft neurogram

        % Scaling method used by Hines and Harte (Speech Comm 2010,
    2012)
```

```
191         % scl_mr = 255/max(max(neurogram_mr_ref(:,1:ind_mr)));
192         % scl_ft = 255/max(max(neurogram_ft_ref(:,1:ind_ft)));
193
194         % New scaling method developed by M. R. Wirtzfeld (see
    Wirtzfeld et al., JARO 2017)
195         scl_mr = 1/50/t_mr(2);
196         scl_ft = 1/50/t_ft(2);
197
198         [NSIM_MR_4(lp,i), ssim_mr] = mssim_ians(scl_mr*
    neurogram_mr_ref(:,1:ind_mr), scl_mr*neurogram_mr_test(:,1:ind_mr
    ), weights, win3x3, window_type );
199         [NSIM_FT_4(lp,i), ssim_ft] = mssim_ians(scl_ft*
    neurogram_ft_ref(:,1:ind_ft), scl_ft*neurogram_ft_test(:,1:ind_ft
    ), weights, win3x3, window_type );
200     end
201 end
202 nsim4mean_mr=mean(NSIM_MR_4,2);
203 nsim4std_mr=std(NSIM_MR_4,1,2);
204 nsim4mean_ft=mean(NSIM_FT_4,2);
205 nsim4std_ft=std(NSIM_FT_4,1,2);
206
207 % Speech-TFS + WGN-ENV
208 for lp = 1:numoffilter
209     for i=1:50
210         d=dir(['TFS_WGN_E_' num2str(filter_n(lp)) 'filter' '*.mat'])
211         load([d(i).name])
212         str=d(i).name;
213         ori_str=extractAfter(str,"filter_")
214         load([ori_str])
```

```
215         teststim=added_filter_bands';
216         refstim=x;
217         Fs_stim=FS;
218         refstim = refstim/rms(refstim)*20e-6*10^(stimdb/20);
219         teststim = teststim/rms(teststim)*20e-6*10^(stimdb/20);
220
221         [neurogram_ft_ref,neurogram_mr_ref,neurogram_Sout_ref
    ,~,~,~,~] = generate_neurogram_function(refstim,Fs_stim,species,
    ag_fs_ref,ag_dbloss_ref);
222         [neurogram_ft_test,neurogram_mr_test,neurogram_Sout_test,
    t_ft,t_mr,t_Sout,CFs] = generate_neurogram_function(teststim,
    Fs_stim,species,ag_fs_test,ag_dbloss_test);
223
224         response_endtime = length(refstim)/Fs_stim+10e-3; %
    Calculate end time of the neural response to the stimulus
225         [tmp, ind_mr] = min(abs(t_mr-response_endtime)); % Find the
    corresponding time index in the mr neurogram
226         [tmp, ind_ft] = min(abs(t_ft-response_endtime)); % Find the
    corresponding time index in the ft neurogram
227
228         % Scaling method used by Hines and Harte (Speech Comm 2010,
    2012)
229         % scl_mr = 255/max(max(neurogram_mr_ref(:,1:ind_mr)));
230         % scl_ft = 255/max(max(neurogram_ft_ref(:,1:ind_ft)));
231
232         % New scaling method developed by M. R. Wirtzfeld (see
    Wirtzfeld et al., JARO 2017)
233         scl_mr = 1/50/t_mr(2);
234         scl_ft = 1/50/t_ft(2);
```

```
235
236         [NSIM_MR_5(lp,i), ssim_mr] = mssim_ians(scl_mr*
    neurogram_mr_ref(:,1:ind_mr), scl_mr*neurogram_mr_test(:,1:ind_mr
    ), weights, win3x3, window_type );
237         [NSIM_FT_5(lp,i), ssim_ft] = mssim_ians(scl_ft*
    neurogram_ft_ref(:,1:ind_ft), scl_ft*neurogram_ft_test(:,1:ind_ft
    ), weights, win3x3, window_type );
238      end
239 end
240 nsim5mean_mr=mean(NSIM_MR_5,2);
241 nsim5std_mr=std(NSIM_MR_5,1,2);
242 nsim5mean_ft=mean(NSIM_FT_5,2);
243 nsim5std_ft=std(NSIM_FT_5,1,2);
244
245 figure (1)
246
247 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
248    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
249    'Speech-TFS + WGN-ENV'};
250
251 errorbar(filter_n,nsim1mean_mr,nsim1std_mr/sqrt(50),'r','linewidth'
    ,2.0)
252 hold on
253 errorbar(filter_n,nsim2mean_mr,nsim2std_mr/sqrt(50),'b','linewidth'
    ,2.0)
254 hold on
255 errorbar(filter_n,nsim3mean_mr,nsim3std_mr/sqrt(50),'m','linewidth'
    ,2.0)
256 hold on
```

```
257 errorbar(filter_n,nsim4mean_mr,nsim4std_mr/sqrt(50),'g','linewidth'
      ,2.0)
258 hold on
259 errorbar(filter_n,nsim5mean_mr,nsim5std_mr/sqrt(50),'c','linewidth'
      ,2.0)
260 ylim([0 1])
261 xlabel('Number of Vocoder Filters')
262 ylabel('Mean-rate NSIM')
263 xlim([0.9 34])
264 set(gca,'XTick',filter_n)
265 set(gca,'xscale','log')
266 grid on
267 legend(casestrs([1 2 3 4 5]),'location','NorthEast')
268
269 figure (2)
270
271 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
272    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
273    'Speech-TFS + WGN-ENV'};
274
275 errorbar(filter_n,nsim1mean_ft,nsim1std_ft/sqrt(50),'r','linewidth'
      ,2.0)
276 hold on
277 errorbar(filter_n,nsim2mean_ft,nsim2std_ft/sqrt(50),'b','linewidth'
      ,2.0)
278 hold on
279 errorbar(filter_n,nsim3mean_ft,nsim3std_ft/sqrt(50),'m','linewidth'
      ,2.0)
280 hold on
```

```
errorbar(filter_n,nsim4mean_ft,nsim4std_ft/sqrt(50),'g','linewidth',2.0)
hold on
errorbar(filter_n,nsim5mean_ft,nsim5std_ft/sqrt(50),'c','linewidth',2.0)
ylim([0 1])
xlabel('Number of Vocoder Filters')
ylabel('Fine-timing NSIM')
xlim([0.9 34])
set(gca,'XTick',filter_n)
set(gca,'xscale','log')
grid on
legend(casestrs([1 2 3 4 5]),'location','NorthEast')
```

B.3 CCC code

```
function [CrossCorr_env,CrossCorr_tfs] = generate_crosscorr(stim_A,stim_B,cf,numfibers,Fs_stim,species,ag_fs,ag_dbloss)
if exist('parfor','builtin') % check if the Matlab Parallel Computation Toolb ox is installed and use appropriate function
    generate_neurogram_function = @generate_spiketrain_BEZ2018_parallelized_test;
    disp('Using parallelized version of neurogram generation function')
else
    generate_neurogram_function = @generate_spiketrain_BEZ2018_test;
    disp('Using serial version of neurogram generation function')
end

```

```
[ neurogram_ft_A ,t_ft_A , CFs ] = generate_neurogram_function (
    numfibers,cf,stim_A,Fs_stim,species,ag_fs,ag_dbloss);
binWidth = t_ft_A(2)-t_ft_A(1);
binEdges = -0.1: binWidth :0.1;

[SAC_A , uniqueFibres_A ] = generate_SAC_ian( neurogram_ft_A ,
    t_ft_A , binEdges );
D_A=t_ft_A(end);
nk_A=size(neurogram_ft_A,1);
k_A=0;
for kk=1:nk_A
    pks= findpeaks(neurogram_ft_A(kk,:));
    k_A=k_A+size(pks,2);
end
r_A=round((k_A/nk_A)/D_A); %average discharged rate
SAC_A = SAC_A /( uniqueFibres_A *(uniqueFibres_A -1) *r_A*r_A*D_A*
    binWidth );
[ neurogram_ft_inv_A , t_ft_inv_A , CFs_inv_A ] =
    generate_neurogram_function (numfibers ,cf , -stim_A , Fs_stim ,
    species ,ag_fs , ag_dbloss );
[ SCC_A ] = generate_SCC_ian( neurogram_ft_A , t_ft_A ,
    neurogram_ft_inv_A , t_ft_inv_A , binEdges );
SCC_A = SCC_A /( numfibers * numfibers *r_A*r_A* D_A * binWidth );

[ neurogram_ft_B ,t_ft_B , CFs_B ] = generate_neurogram_function (
    numfibers ,cf , stim_B , Fs_stim , species ,ag_fs , ag_dbloss );
binWidth = t_ft_B(2)-t_ft_B(1);
binEdges = -0.1: binWidth :0.1;
```

```
31 [SAC_B , uniqueFibres_B ] = generate_SAC_ian( neurogram_ft_B ,
       t_ft_B , binEdges );
32 D_B=t_ft_B(end);
33 nk_B=size(neurogram_ft_B,1);
34 k_B=0;
35 for kk=1:nk_B
36     pks= findpeaks(neurogram_ft_B(kk,:));
37     k_B=k_B+size(pks,2);
38 end
39 r_B=round((k_B/nk_B)/D_B); %average discharged rate
40 SAC_B = SAC_B /( uniqueFibres_B *( uniqueFibres_B -1) *r_B*r_B*D_B*
       binWidth );
41
42 [ neurogram_ft_inv_B , t_ft_inv_B , CFs_inv_B ] =
       generate_neurogram_function (numfibers ,cf , -stim_B , Fs_stim ,
       species ,ag_fs , ag_dbloss );
43 [ SCC_B ] = generate_SCC_ian( neurogram_ft_B , t_ft_B ,
       neurogram_ft_inv_B , t_ft_inv_B , binEdges );
44 SCC_B = SCC_B /( numfibers *numfibers*r_B*r_B*D_B*binWidth );
45
46 [ SAC_AB_temp1 ] = generate_SCC_ian( neurogram_ft_A , t_ft_A ,
       neurogram_ft_B , t_ft_B , binEdges );
47 [ SAC_AB_temp2 ] = generate_SCC_ian( neurogram_ft_inv_A , t_ft_inv_A
        , neurogram_ft_inv_B , t_ft_inv_B , binEdges );
48 SAC_AB = ( SAC_AB_temp1 + SAC_AB_temp2 ) /2;
49 D=min([D_A D_B]);
50 SAC_AB = SAC_AB /( numfibers * numfibers *r_A*r_B*D*binWidth );
51
```

```
[ SCC_AB_temp1 ] = generate_SCC_ian( neurogram_ft_A , t_ft_A ,
    neurogram_ft_inv_B , t_ft_inv_B , binEdges );
[ SCC_AB_temp2 ] = generate_SCC_ian( neurogram_ft_inv_A , t_ft_inv_A
     , neurogram_ft_B , t_ft_B , binEdges );
SCC_AB = ( SCC_AB_temp1 + SCC_AB_temp2 ) /2;
SCC_AB = SCC_AB /( numfibers * numfibers *r_A*r_B*D*binWidth );

clear neurogram*
difcor_A = SAC_A - SCC_A;
sumcor_A = ( SAC_A + SCC_A )/2;
difcor_B = SAC_B - SCC_B;
sumcor_B = ( SAC_B + SCC_B )/2;
difcor_AB = SAC_AB - SCC_AB;
sumcor_AB = ( SAC_AB + SCC_AB )/2;

fch=64;
fs=1/binWidth;
[B,A] = butter (2 ,fch/(fs/2));   % just lowpass filter

sumcor_A = filtfilt (B,A, sumcor_A );
% difcor_A = filtfilt (B,A, difcor_A );
sumcor_B = filtfilt (B,A, sumcor_B );
% difcor_B = filtfilt (B,A, difcor_B );
sumcor_AB = filtfilt (B,A, sumcor_AB );
% difcor_AB = filtfilt (B,A, difcor_AB );
CrossCorr_env = (max(sumcor_AB(9500:10500) -1) )/ sqrt ( max(
    sumcor_A(9500:10500)-1 )* max (sumcor_B(9500:10500) -1 ));
CrossCorr_tfs = (max(difcor_AB ))/ sqrt ( max (difcor_A)* max (
    difcor_B));
```

```
% Check to see if running under Matlab or Octave
if exist ('OCTAVE_VERSION', 'builtin') ~= 0
pkg load signal ;
if exist ('rms') <1
rms = @(x) sqrt(mean(x.^2));
end
end
if exist('parfor','builtin') % check if the Matlab Parallel
    Computation Toolbox is installed and use appropriate function
     generate_neurogram_function =
    @generate_spiketrain_BEZ2018_parallelized_test;
     disp('Using parallelized version of neurogram generation
    function')
else
     generate_neurogram_function = @generate_spiketrain_BEZ2018_test;
     disp('Using serial version of neurogram generation function')
end

% Set audiogram
ag_fs = [125 250 500 1e3 2e3 4e3 8e3];
ag_dbloss = [0 0 0 0 0 0 0]; % Normal hearing
numfibers = 25;
cf =550;
species = 2;
Fs_stim = 16e3;
Fm_stim_A=10
Fm_stim_B=2:2:20;
for j=1:1:length(Fm_stim_B)
     t = 0:1/Fs_stim:10;
```

```
27      x_ta=sin(2*pi*Fm_stim_A*t);
28      x_tb=sin(2*pi*Fm_stim_B(j)*t);
29      A=1;
30      B=1;
31      ka=1;
32      kb=1;
33      Fc_stim_A=500;
34      Fc_stim_B=500;
35      stim_A = A*(1+ka*x_ta).*cos(2*pi*Fc_stim_A*t);
36      stim_B = B*(1+kb*x_tb).*cos(2*pi*Fc_stim_B*t);
37
38      stimdb_A = 35; % speech level in dB SPL
39      stim_A = stim_A/rms(stim_A)*20e-6*10^(stimdb_A /20);
40      stimdb_B = 35; % speech level in dB SPL
41      stim_B = stim_B/rms(stim_B)*20e-6*10^(stimdb_B /20);
42      [ neurogram_ft_A ,t_ft_A , CFs ] = generate_neurogram_function(
    numfibers ,cf , stim_A , Fs_stim , species ,ag_fs , ag_dbloss );
43      binWidth = 50e-6;
44      binEdges = -0.1: binWidth :0.1;
45      [SAC_A , uniqueFibres_A ] = generate_SAC_ian( neurogram_ft_A ,
    t_ft_A , binEdges );
46      D_A=t_ft_A(end);
47      nk_A=size(neurogram_ft_A,1);
48      k_A=0;
49      for kk=1:nk_A
50          pks= findpeaks(neurogram_ft_A(kk,:));
51          k_A=k_A+size(pks,2);
52      end
53      r_A=round((k_A/nk_A)/D_A); %average discharged rate
```

```
54      SAC_A = SAC_A /( uniqueFibres_A *(uniqueFibres_A -1) *r_A*r_A*
    D_A* binWidth );
55      [ neurogram_ft_inv_A , t_ft_inv_A , CFs_inv_A ] =
    generate_neurogram_function(numfibers ,cf , -stim_A , Fs_stim ,
    species ,ag_fs , ag_dbloss );
56      [ SCC_A ] = generate_SCC_ian( neurogram_ft_A , t_ft_A ,
    neurogram_ft_inv_A , t_ft_inv_A , binEdges );
57      SCC_A = SCC_A /(numfibers * numfibers *r_A*r_A* D_A * binWidth);
58      timeSeries = binEdges(1:end -1)+binWidth/2;
59
60      [ neurogram_ft_B ,t_ft_B , CFs_B ] = generate_neurogram_function
    (numfibers ,cf , stim_B , Fs_stim , species ,ag_fs , ag_dbloss );
61      binWidth = 50e-6;
62      binEdges = -0.1: binWidth :0.1;
63  %     binEdges = -0.025: binWidth :0.025;
64      [SAC_B , uniqueFibres_B ] = generate_SAC_ian( neurogram_ft_B ,
    t_ft_B , binEdges );
65      D_B=t_ft_B(end);
66      nk_B=size(neurogram_ft_B ,1);
67      k_B=0;
68      for kk=1:nk_B
69          pks= findpeaks(neurogram_ft_B(kk,:));
70          k_B=k_B+size(pks ,2);
71      end
72      r_B=round((k_B/nk_B)/D_B); %average discharged rate
73      SAC_B = SAC_B /( uniqueFibres_B *( uniqueFibres_B -1) *r_B*r_B*
    D_B*binWidth );
```

```
74  [ neurogram_ft_inv_B , t_ft_inv_B , CFs_inv_B ] =
    generate_neurogram_function (numfibers ,cf , -stim_B , Fs_stim ,
    species ,ag_fs , ag_dbloss );
75  [ SCC_B ] = generate_SCC_ian( neurogram_ft_B , t_ft_B ,
    neurogram_ft_inv_B , t_ft_inv_B , binEdges );
76  SCC_B = SCC_B /( numfibers *numfibers*r_B*r_B*D_B*binWidth );
77  [ SAC_AB_temp1 ] = generate_SCC_ian( neurogram_ft_A , t_ft_A ,
    neurogram_ft_B , t_ft_B , binEdges );
78  [ SAC_AB_temp2 ] = generate_SCC_ian( neurogram_ft_inv_A ,
    t_ft_inv_A , neurogram_ft_inv_B , t_ft_inv_B , binEdges );
79  SAC_AB = ( SAC_AB_temp1 + SAC_AB_temp2 ) /2;
80  D=min([D_A D_B]);
81  SAC_AB = SAC_AB /( numfibers * numfibers *r_A*r_B*D*binWidth );
82  [ SCC_AB_temp1 ] = generate_SCC_ian( neurogram_ft_A , t_ft_A ,
    neurogram_ft_inv_B , t_ft_inv_B , binEdges );
83  [ SCC_AB_temp2 ] = generate_SCC_ian( neurogram_ft_inv_A ,
    t_ft_inv_A , neurogram_ft_B , t_ft_B , binEdges );
84  SCC_AB = ( SCC_AB_temp1 + SCC_AB_temp2 ) /2;
85  SCC_AB = SCC_AB /( numfibers * numfibers *r_A*r_B*D*binWidth );
86  clear neurogram*
87  difcor_A = SAC_A - SCC_A;
88  sumcor_A = ( SAC_A + SCC_A )/2;
89  difcor_B = SAC_B - SCC_B;
90  sumcor_B = ( SAC_B + SCC_B )/2;
91  difcor_AB = SAC_AB - SCC_AB;
92  sumcor_AB = ( SAC_AB + SCC_AB )/2;
93  fcl=0.5;
94  fch=64;
95  fs=1/binWidth;
```

```
96      % [B,A] = butter (2 ,[fcl fch]/(fs/2)); % bandpass filter
97      [B,A] = butter (2 ,fch/(fs/2));   % just lowpass filter
98      % filter below 64 butter is ok
99  %     [B,A] = butter (2 ,0.0001 , 'high'); % just highpass filter
100     sumcor_A = filtfilt (B,A, sumcor_A );
101 %     difcor_A = filtfilt (B,A, difcor_A );
102     sumcor_B = filtfilt (B,A, sumcor_B );
103 %     difcor_B = filtfilt (B,A, difcor_B );
104     sumcor_AB = filtfilt (B,A, sumcor_AB );
105 %     difcor_AB = filtfilt (B,A, difcor_AB );
106     % CrossCorr_env(j) = (max( sumcor_AB -1) )/ sqrt ( max(sumcor_A
    (1500:2500) -1)* max (sumcor_B -1));
107     CrossCorr_env(j) = (max(sumcor_AB(1500:2500) -1) )/ sqrt ( max(
    sumcor_A(1500:2500) -1 )* max (sumcor_B(1500:2500) -1 ));
108     CrossCorr_tfs(j) = (max( difcor_AB ))/ sqrt ( max (difcor_A)*
    max (difcor_B));
109 end
110
111 %%
112
113 figure
114
115 subplot(2,1,1)
116 plot(Fm_stim_B,CrossCorr_env)
117 xlabel('Modulation Frequency (Hz)')
118 ylabel('CrossCorr-env')
119 ylim([0 1.1])
120 title('SPL 65dB (a)')
121
```

```
122 subplot(2,1,2)
123 plot(Fm_stim_B,CrossCorr_tfs)
124 xlabel('Modulation Frequency (Hz)')
125 ylabel('CrossCorr-tfs')
126 ylim([0 1.1])
127 title('SPL 65dB (b)')
```

```
1 % Check to see if running under Matlab or Octave
2 if exist ('OCTAVE_VERSION', 'builtin') ~= 0
3 pkg load signal ;
4 if exist ('rms') <1
5 rms = @(x) sqrt(mean(x.^2));
6 end
7 end
8 if exist('parfor','builtin') % check if the Matlab Parallel
    Computation Toolbox is installed and use appropriate function
9     generate_neurogram_function =
    @generate_spiketrain_BEZ2018_parallelized_test;
10     disp('Using parallelized version of neurogram generation
    function')
11 else
12     generate_neurogram_function = @generate_spiketrain_BEZ2018_test;
13     disp('Using serial version of neurogram generation function')
14 end
15
16 % Set audiogram
17 ag_fs = [125 250 500 1e3 2e3 4e3 8e3];
18 ag_dbloss = [0 0 0 0 0 0 0]; % Normal hearing
19 numfibers = 25;
20 cf =550;
```

```
21 species = 2;
22
23 Fs_stim = 16e3;
24 Fc_stim_A=1000;
25 Fc_stim_B=800:50:1200;
26 for j=1:1:length(Fc_stim_B)
27     Fm_stim_A=10
28     Fm_stim_B=10;
29     t = 0:1/Fs_stim:10;
30     x_ta=sin(2*pi*Fm_stim_A*t);
31     x_tb=sin(2*pi*Fm_stim_B*t);
32     A=1;
33     B=1;
34     ka=1;
35     kb=1;
36     stim_A = A*(1+ka*x_ta).*cos(2*pi*Fc_stim_A*t);
37     stim_B = B*(1+kb*x_tb).*cos(2*pi*Fc_stim_B(j)*t);
38
39     stimdb_A = 65; % speech level in dB SPL
40     stim_A = stim_A/rms(stim_A)*20e-6*10^(stimdb_A /20);
41     stimdb_B = 65; % speech level in dB SPL
42     stim_B = stim_B/rms(stim_B)*20e-6*10^(stimdb_B /20);
43     [ neurogram_ft_A ,t_ft_A , CFs ] = generate_neurogram_function(
    numfibers ,cf , stim_A , Fs_stim , species ,ag_fs , ag_dbloss );
44     binWidth = 50e-6;
45     binEdges = -0.1: binWidth :0.1;
46     [SAC_A , uniqueFibres_A ] = generate_SAC_ian( neurogram_ft_A ,
    t_ft_A , binEdges );
47     D_A=t_ft_A(end);
```

```
48      nk_A=size(neurogram_ft_A,1);
49      k_A=0;
50      for kk=1:nk_A
51          pks= findpeaks(neurogram_ft_A(kk,:));
52          k_A=k_A+size(pks,2);
53      end
54      r_A=round((k_A/nk_A)/D_A); %average discharged rate
55      SAC_A = SAC_A /( uniqueFibres_A *(uniqueFibres_A -1) *r_A*r_A*
    D_A* binWidth );
56      [ neurogram_ft_inv_A , t_ft_inv_A , CFs_inv_A ] =
    generate_neurogram_function(numfibers ,cf , -stim_A , Fs_stim ,
    species ,ag_fs , ag_dbloss );
57      [ SCC_A ] = generate_SCC_ian( neurogram_ft_A , t_ft_A ,
    neurogram_ft_inv_A , t_ft_inv_A , binEdges );
58      SCC_A = SCC_A /(numfibers * numfibers *r_A*r_A* D_A * binWidth);
59      timeSeries = binEdges(1:end -1)+binWidth/2;
60
61      [ neurogram_ft_B ,t_ft_B , CFs_B ] = generate_neurogram_function
    (numfibers ,cf , stim_B , Fs_stim , species ,ag_fs , ag_dbloss );
62      binWidth = 50e-6;
63      binEdges = -0.1: binWidth :0.1;
64      [SAC_B , uniqueFibres_B ] = generate_SAC_ian( neurogram_ft_B ,
    t_ft_B , binEdges );
65      D_B=t_ft_B(end);
66      nk_B=size(neurogram_ft_B,1);
67      k_B=0;
68      for kk=1:nk_B
69          pks= findpeaks(neurogram_ft_B(kk,:));
70          k_B=k_B+size(pks,2);
```

```
71      end
72      r_B=round((k_B/nk_B)/D_B); %average discharged rate
73      SAC_B = SAC_B /( uniqueFibres_B *( uniqueFibres_B -1) *r_B*r_B*
        D_B*binWidth );
74      [ neurogram_ft_inv_B , t_ft_inv_B , CFs_inv_B ] =
        generate_neurogram_function (numfibers ,cf , -stim_B , Fs_stim ,
        species ,ag_fs , ag_dbloss );
75      [ SCC_B ] = generate_SCC_ian( neurogram_ft_B , t_ft_B ,
        neurogram_ft_inv_B , t_ft_inv_B , binEdges );
76      SCC_B = SCC_B /( numfibers *numfibers*r_B*r_B*D_B*binWidth );
77      [ SAC_AB_temp1 ] = generate_SCC_ian( neurogram_ft_A , t_ft_A ,
        neurogram_ft_B , t_ft_B , binEdges );
78      [ SAC_AB_temp2 ] = generate_SCC_ian( neurogram_ft_inv_A ,
        t_ft_inv_A , neurogram_ft_inv_B , t_ft_inv_B , binEdges );
79      SAC_AB = ( SAC_AB_temp1 + SAC_AB_temp2 ) /2;
80      D=min([D_A D_B]);
81      SAC_AB = SAC_AB /( numfibers * numfibers *r_A*r_B*D*binWidth );
82      [ SCC_AB_temp1 ] = generate_SCC_ian( neurogram_ft_A , t_ft_A ,
        neurogram_ft_inv_B , t_ft_inv_B , binEdges );
83      [ SCC_AB_temp2 ] = generate_SCC_ian( neurogram_ft_inv_A ,
        t_ft_inv_A , neurogram_ft_B , t_ft_B , binEdges );
84      SCC_AB = ( SCC_AB_temp1 + SCC_AB_temp2 ) /2;
85      SCC_AB = SCC_AB /( numfibers * numfibers *r_A*r_B*D*binWidth );
86      clear neurogram*
87      difcor_A = SAC_A - SCC_A;
88      sumcor_A = ( SAC_A + SCC_A )/2;
89      difcor_B = SAC_B - SCC_B;
90      sumcor_B = ( SAC_B + SCC_B )/2;
91      difcor_AB = SAC_AB - SCC_AB;
```

```
92      sumcor_AB = ( SAC_AB + SCC_AB )/2;
93      fcl=0.5;
94      fch=64;
95      fs=1/binWidth;
96      [B,A] = butter (2 ,fch/(fs/2));   % just lowpass filter
97  %     [B,A] = butter (2 ,[fcl fch]/(fs/2));
98  %     [B,A] = butter (2 ,0.0001 , 'high');
99      sumcor_A = filtfilt (B,A, sumcor_A );
100 %     difcor_A = filtfilt (B,A, difcor_A );
101     sumcor_B = filtfilt (B,A, sumcor_B );
102 %     difcor_B = filtfilt (B,A, difcor_B );
103     sumcor_AB = filtfilt (B,A, sumcor_AB );
104 %     difcor_AB = filtfilt (B,A, difcor_AB );
105 %     CrossCorr_env(j) = (max(sumcor_AB(1500:2500) -1) )/ sqrt ( max
    ( sumcor_A(1500:2500) -1 )* max (sumcor_B(1500:2500) -1 ));
106     CrossCorr_env(j) = (max(sumcor_AB(1500:2500) -1) )/ sqrt ( max(
    sumcor_A(1500:2500)-1 )* max (sumcor_B(1500:2500) -1 ));
107     CrossCorr_tfs(j) = (max(difcor_AB ))/ sqrt ( max (difcor_A)* max
     (difcor_B));
108 end
109
110 figure
111 subplot(2,1,1)
112 plot(Fc_stim_B,CrossCorr_env)
113 xlabel('Carrier Frequency (Hz)')
114 ylabel('CrossCorr-env')
115 ylim([0 1.1])
116 title('SPL 65dB (a)')
117
```

```
118 subplot(2,1,2)
119 plot(Fc_stim_B,CrossCorr_tfs)
120 xlabel('Carrier Frequency (Hz)')
121 ylabel('CrossCorr-tfs')
122 ylim([0 1.1])
123 title('SPL 65dB (b)')
```

```
1 % Check to see if running under Matlab or Octave
2 if exist ('OCTAVE_VERSION', 'builtin') ~= 0
3 pkg load signal ;
4 if exist ('rms') <1
5 rms = @(x) sqrt(mean(x.^2));
6 end
7 end
8
9 % Set audiogram
10 ag_fs = [125 250 500 1e3 2e3 4e3 8e3];
11 ag_dbloss = [0 0 0 0 0 0 0]; % Normal hearing
12 numfibers = 10;
13 cf =logspace(log10(200),log10(8e3),10);
14 numcfs = length(cf);
15 cf_ind=find(cf<=2.5e3);
16
17 numwords = 50;
18
19 species = 2; % Human cochlear tuning ( Shera et al., 2002)
20
21 load('FS.mat')
22 filter_n = [1 2 3 6 8 16 32];
23
```

```
numoffilter = length(filter_n);

CrossCorr1_env=zeros(numoffilter,numwords,numcfs);
CrossCorr2_env=zeros(numoffilter,numwords,numcfs);
CrossCorr3_env=zeros(numoffilter,numwords,numcfs);
CrossCorr4_env=zeros(numoffilter,numwords,numcfs);
CrossCorr5_env=zeros(numoffilter,numwords,numcfs);
CrossCorr1_tfs=zeros(numoffilter,numwords,numcfs);
CrossCorr2_tfs=zeros(numoffilter,numwords,numcfs);
CrossCorr3_tfs=zeros(numoffilter,numwords,numcfs);
CrossCorr4_tfs=zeros(numoffilter,numwords,numcfs);
CrossCorr5_tfs=zeros(numoffilter,numwords,numcfs);

% Speech-ENV + MN-TFS
for lp = 1:numoffilter
    for ii=1:numwords
        d=dir(['E_matched_noise_TFS' num2str(filter_n(lp)) 'filters'
    '*.mat'])
        load([d(ii).name])
        str=d(ii).name;
        ori_str=extractAfter(str,"filters_")
        load([ori_str])
        Fs_stim=FS;
        stim_A = x;
        stim_B = added_filter_bands';
        stimdb_A = 65; % speech level in dB SPL
        stim_A = stim_A/rms(stim_A)*20e-6*10^(stimdb_A /20);
        stimdb_B = 65; % speech level in dB SPL
        stim_B = stim_B/rms(stim_B)*20e-6*10^(stimdb_B /20);
```

```
52          for jj = 1:length(cf)
53              [CrossCorr1_env(lp,ii,jj),CrossCorr1_tfs(lp,ii,jj)] =
    generate_crosscorr(stim_A,stim_B,cf(jj),numfibers,Fs_stim,species
    ,ag_fs,ag_dbloss);
54          end
55      end
56  end
57
58  CrossCorr1_envmean=mean(CrossCorr1_env,3);
59  CrossCorr1_tfsmean=mean(CrossCorr1_tfs(:,:,1:length(cf_ind)),3);
60  CrossCorr1_envavg=mean(CrossCorr1_envmean,2);
61  CrossCorr1_envstd=std(CrossCorr1_envmean,1,2);
62  CrossCorr1_tfsavg=mean(CrossCorr1_tfsmean,2);
63  CrossCorr1_tfsstd=std(CrossCorr1_tfsmean,1,2);
64
65  % Speech-ENV + WGN-TFS
66  for lp = 1:numoffilter
67      for ii=1:numwords
68          d=dir(['E_WGN_TFS_' num2str(filter_n(lp)) 'filter' '*.mat'])
69          load([d(ii).name])
70          str=d(ii).name;
71          ori_str=extractAfter(str,"filter_")
72          load([ori_str])
73          Fs_stim=FS;
74          stim_A = x;
75          stim_B = added_filter_bands';
76          stimdb_A = 65; % speech level in dB SPL
77          stim_A = stim_A/rms(stim_A)*20e-6*10^(stimdb_A /20);
78          stimdb_B = 65; % speech level in dB SPL
```

```
79          stim_B = stim_B/rms(stim_B)*20e-6*10^(stimdb_B /20);
80          for jj = 1:length(cf)
81              [CrossCorr2_env(lp,ii,jj),CrossCorr2_tfs(lp,ii,jj)] =
    generate_crosscorr(stim_A,stim_B,cf(jj),numfibers,Fs_stim,species
    ,ag_fs,ag_dbloss);
82          end
83      end
84  end
85  CrossCorr2_envmean=mean(CrossCorr2_env,3);
86  CrossCorr2_tfsmean=mean(CrossCorr2_tfs(:,:,1:length(cf_ind)),3);
87  CrossCorr2_envavg=mean(CrossCorr2_envmean,2);
88  CrossCorr2_envstd=std(CrossCorr2_envmean,1,2);
89  CrossCorr2_tfsavg=mean(CrossCorr2_tfsmean,2);
90  CrossCorr2_tfsstd=std(CrossCorr2_tfsmean,1,2);
91
92  % Speech-TFS + MN-ENV
93  for lp = 1:numoffilter
94      for ii=1:numwords
95          d=dir(['TFS_matched_noise_E' num2str(filter_n(lp)) 'filters'
    '*.mat'])
96          load([d(ii).name])
97          str=d(ii).name;
98          ori_str=extractAfter(str,"filters_")
99          load([ori_str])
100         Fs_stim=FS;
101         stim_A = x;
102         stim_B = added_filter_bands';
103         stimdb_A = 65; % speech level in dB SPL
104         stim_A = stim_A/rms(stim_A)*20e-6*10^(stimdb_A /20);
```

```
105         stimdb_B = 65; % speech level in dB SPL
106         stim_B = stim_B/rms(stim_B)*20e-6*10^(stimdb_B /20);
107         for jj = 1:length(cf)
108             [CrossCorr3_env(lp,ii,jj),CrossCorr3_tfs(lp,ii,jj)] =
    generate_crosscorr(stim_A,stim_B,cf(jj),numfibers,Fs_stim,species
    ,ag_fs,ag_dbloss);
109         end
110     end
111 end
112 CrossCorr3_envmean=mean(CrossCorr3_env,3);
113 CrossCorr3_tfsmean=mean(CrossCorr3_tfs(:,:,1:length(cf_ind)),3);
114 CrossCorr3_envavg=mean(CrossCorr3_envmean,2);
115 CrossCorr3_envstd=std(CrossCorr3_envmean,1,2);
116 CrossCorr3_tfsavg=mean(CrossCorr3_tfsmean,2);
117 CrossCorr3_tfsstd=std(CrossCorr3_tfsmean,1,2);
118
119 % Speech-TFS Only
120 for lp = 1:numoffilter
121     for ii=1:numwords
122         d=dir(['TFS_only_' num2str(filter_n(lp)) 'filters' '*.mat'])
123         load([d(ii).name])
124         str=d(ii).name;
125         ori_str=extractAfter(str,"filters_")
126         load([ori_str])
127         Fs_stim=FS;
128         stim_A = x;
129         stim_B = added_filter_bands';
130         stimdb_A = 65; % speech level in dB SPL
131         stim_A = stim_A/rms(stim_A)*20e-6*10^(stimdb_A /20);
```

```
132         stimdb_B = 65; % speech level in dB SPL
133         stim_B = stim_B/rms(stim_B)*20e-6*10^(stimdb_B /20);
134         for jj = 1:length(cf)
135             [CrossCorr4_env(lp,ii,jj),CrossCorr4_tfs(lp,ii,jj)] =
    generate_crosscorr(stim_A,stim_B,cf(jj),numfibers,Fs_stim,species
    ,ag_fs,ag_dbloss);
136         end
137     end
138 end
139 CrossCorr4_envmean=mean(CrossCorr4_env,3);
140 CrossCorr4_tfsmean=mean(CrossCorr4_tfs(:,:,1:length(cf_ind)),3);
141 CrossCorr4_envavg=mean(CrossCorr4_envmean,2);
142 CrossCorr4_envstd=std(CrossCorr4_envmean,1,2);
143 CrossCorr4_tfsavg=mean(CrossCorr4_tfsmean,2);
144 CrossCorr4_tfsstd=std(CrossCorr4_tfsmean,1,2);
145
146 % Speech-TFS + WGN-ENV
147 for lp = 1:numoffilter
148     for ii=1:numwords
149         d=dir(['TFS_WGN_E_' num2str(filter_n(lp)) 'filter' '*.mat'])
150         load([d(ii).name])
151         str=d(ii).name;
152         ori_str=extractAfter(str,"filter_")
153         load([ori_str])
154         Fs_stim=FS;
155         stim_A = x;
156         stim_B = added_filter_bands';
157         stimdb_A = 65; % speech level in dB SPL
158         stim_A = stim_A/rms(stim_A)*20e-6*10^(stimdb_A /20);
```

```
159         stimdb_B = 65; % speech level in dB SPL
160         stim_B = stim_B/rms(stim_B)*20e-6*10^(stimdb_B /20);
161         for jj = 1:numcfs
162             [CrossCorr5_env(lp,ii,jj),CrossCorr5_tfs(lp,ii,jj)] =
    generate_crosscorr(stim_A,stim_B,cf(jj),numfibers,Fs_stim,species
    ,ag_fs,ag_dbloss);
163         end
164     end
165 end
166 CrossCorr5_envmean=mean(CrossCorr5_env,3);
167 CrossCorr5_tfsmean=mean(CrossCorr5_tfs(:,:,1:length(cf_ind)),3);
168 CrossCorr5_envavg=mean(CrossCorr5_envmean,2);
169 CrossCorr5_envstd=std(CrossCorr5_envmean,1,2);
170 CrossCorr5_tfsavg=mean(CrossCorr5_tfsmean,2);
171 CrossCorr5_tfsstd=std(CrossCorr5_tfsmean,1,2);
172
173 figure (1)
174 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
175     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
176     'Speech-TFS + WGN-ENV'};
177
178 errorbar(filter_n,CrossCorr1_envavg,CrossCorr1_envstd/sqrt(50),'r','
    linewidth',2.0)
179 hold on
180 errorbar(filter_n,CrossCorr2_envavg,CrossCorr2_envstd/sqrt(50),'b','
    linewidth',2.0)
181 hold on
182 errorbar(filter_n,CrossCorr3_envavg,CrossCorr3_envstd/sqrt(50),'m','
    linewidth',2.0)
```

```
183 hold on
184 errorbar(filter_n,CrossCorr4_envavg,CrossCorr4_envstd/sqrt(50),'g','
    linewidth',2.0)
185 hold on
186 errorbar(filter_n,CrossCorr5_envavg,CrossCorr5_envstd/sqrt(50),'c','
    linewidth',2.0)
187 ylim([0 1])
188 xlabel('Number of Vocoder Filters')
189 ylabel('CrossCorr-env')
190 xlim([0.9 34])
191 set(gca,'XTick',filter_n)
192 set(gca,'xscale','log')
193 grid on
194 legend(casestrs([1 2 3 4 5]),'location','SouthEast')
195
196 figure (2)
197 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
198     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
199     'Speech-TFS + WGN-ENV'};
200
201 errorbar(filter_n,CrossCorr1_tfsavg,CrossCorr1_tfsstd/sqrt(50),'r','
    linewidth',2.0)
202 hold on
203 errorbar(filter_n,CrossCorr2_tfsavg,CrossCorr2_tfsstd/sqrt(50),'b','
    linewidth',2.0)
204 hold on
205 errorbar(filter_n,CrossCorr3_tfsavg,CrossCorr3_tfsstd/sqrt(50),'m','
    linewidth',2.0)
206 hold on
```

```
errorbar(filter_n,CrossCorr4_tfsavg,CrossCorr4_tfsstd/sqrt(50),'g','
    linewidth',2.0)
hold on
errorbar(filter_n,CrossCorr5_tfsavg,CrossCorr5_tfsstd/sqrt(50),'c','
    linewidth',2.0)
ylim([0 1])
xlabel('Number of Vocoder Filters')
ylabel('CrossCorr-tfs')
xlim([0.9 34])
set(gca,'XTick',filter_n)
set(gca,'xscale','log')
grid on
legend(casestrs([1 2 3 4 5]),'location','SouthEast')
```

B.4 linear regression code

```
load('STMIresults1_Yujie.mat')
load('NSIMresults1_Yujie.mat')
load('CCC_chimaera_FullResults_7Nov2023.mat')
load('perceptual_identification.mat')
perceptual_res=[sampmeanofint(:,3);sampmeanofint(:,1);sampmeanofint
    (:,4);sampmeanofint(:,5);sampmeanofint(:,2)];
perceptual_rau=rau(perceptual_res);
figure
filter_n = [1 2 3 6 8 16 32];
casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
    'Speech-TFS + WGN-ENV'};

plot(filter_n,perceptual_rau(1:1*7),'r','linewidth',2.0)
```

```
hold on
plot(filter_n,perceptual_rau(1*7+1:2*7),'b','linewidth',2.0)
hold on
plot(filter_n,perceptual_rau(2*7+1:3*7),'m','linewidth',2.0)
hold on
plot(filter_n,perceptual_rau(3*7+1:4*7),'g','linewidth',2.0)
hold on
plot(filter_n,perceptual_rau(4*7+1:5*7),'c','linewidth',2.0)
xlabel('Number of Vocoder Filters')
ylabel('Perceptual Identification')
ylim([0 140])

set(gca,'XTick',filter_n)
set(gca,'xscale','log')
grid on
legend(casestrs([1 2 3 4 5]),'location','SouthEast')
%% STMI only
stmi_res=[100*stmi1mean;100*stmi2mean;100*stmi3mean;100*stmi4mean
    ;100*stmi5mean];
stmi_mdl = fitlm(stmi_res,perceptual_rau)
% stmi_mdl_b0=table2array(stmi_mdl.Coefficients(1,1));
% stmi_mdl_b1=table2array(stmi_mdl.Coefficients(2,1));
% stmi_mdl_x=stmi_mdl_b0+stmi_mdl_b1*stmi_res;
stmi_R2 = stmi_mdl.Rsquared.Adjusted;
stmi_pValue = stmi_mdl.Coefficients.pValue(2);

figure
casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
```

```
42     'Speech-TFS + WGN-ENV'};
43 scatter(stmi_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled",'r')
44 hold on
45 scatter(stmi_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7),"filled
       ",'b')
46 hold on
47 scatter(stmi_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7),"filled
       ",'m')
48 hold on
49 scatter(stmi_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7),"filled
       ",'g')
50 hold on
51 scatter(stmi_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7),"filled
       ",'c')
52 xlabel('STMI-Neural Metric Identification')
53 ylabel('Perceptual Identification')
54 hold on
55 plot(1:140,'k')
56 xlim([0 140])
57 ylim([0 140])
58 grid off
59 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
60 text(5, 100, ...
61       sprintf('Adj. R^2 = %.3f (p-value = %.3f)', stmi_R2,
       stmi_pValue), ...
62       'FontSize', 10);
63
64 figure
65 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
```

```
66      'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
67      'Speech-TFS + WGN-ENV'};
68  filter_n = [1 2 3 6 8 16 32];
69  plot(filter_n,stmi_mdl.Fitted(1:1*7),'r','linewidth',2.0)
70  hold on
71  plot(filter_n,stmi_mdl.Fitted(1*7+1:2*7),'b','linewidth',2.0)
72  hold on
73  plot(filter_n,stmi_mdl.Fitted(2*7+1:3*7),'m','linewidth',2.0)
74  hold on
75  plot(filter_n,stmi_mdl.Fitted(3*7+1:4*7),'g','linewidth',2.0)
76  hold on
77  plot(filter_n,stmi_mdl.Fitted(4*7+1:5*7),'c','linewidth',2.0)
78
79  xlabel('Number of Vocoder Filters')
80  ylabel('STMI-Neural Metric Identification')
81
82  ylim([0 140])
83
84  set(gca,'XTick',filter_n)
85  set(gca,'xscale','log')
86  grid on
87  legend(casestrs([1 2 3 4 5]),'location','SouthWest')
88  %% NSIM FT only
89  nsim_ft_res=[100*nsim1mean_ft;100*nsim2mean_ft;100*nsim3mean_ft;100*
        nsim4mean_ft;100*nsim5mean_ft];
90  nsim_ft_mdl = fitlm(nsim_ft_res,perceptual_rau)
91  % nsim_ft_mdl_b0=table2array(nsim_ft_mdl.Coefficients(1,1));
92  % nsim_ft_mdl_b1=table2array(nsim_ft_mdl.Coefficients(2,1));
93  % nsim_ft_mdl_x=nsim_ft_mdl_b0+nsim_ft_mdl_b1*nsim_ft_res;
```

```
nsim_ft_R2 = nsim_ft_mdl.Rsquared.Adjusted;
nsim_ft_pValue = nsim_ft_mdl.Coefficients.pValue(2);

figure
casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
    'Speech-TFS + WGN-ENV'};
scatter(nsim_ft_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled",'r'
    )
hold on
scatter(nsim_ft_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7),"
    filled",'b')
hold on
scatter(nsim_ft_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7),"
    filled",'m')
hold on
scatter(nsim_ft_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7),"
    filled",'g')
hold on
scatter(nsim_ft_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7),"
    filled",'c')
xlabel('FT NSIM-Neural Metric Identification')
ylabel('Perceptual Identification')
hold on
plot(1:140,'k')
xlim([0 140])
ylim([0 140])
grid off
legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
```

```
118 text(5, 100, ...
119       sprintf('Adj. R^2 = %.3f (p-value = %.3f)', nsim_ft_R2,
      nsim_ft_pValue), ...
120       'FontSize', 10);
121
122 figure
123 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
124    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
125    'Speech-TFS + WGN-ENV'};
126 filter_n = [1 2 3 6 8 16 32];
127 plot(filter_n,nsim_ft_mdl.Fitted(1:1*7),'r','linewidth',2.0)
128 hold on
129 plot(filter_n,nsim_ft_mdl.Fitted(1*7+1:2*7),'b','linewidth',2.0)
130 hold on
131 plot(filter_n,nsim_ft_mdl.Fitted(2*7+1:3*7),'m','linewidth',2.0)
132 hold on
133 plot(filter_n,nsim_ft_mdl.Fitted(3*7+1:4*7),'g','linewidth',2.0)
134 hold on
135 plot(filter_n,nsim_ft_mdl.Fitted(4*7+1:5*7),'c','linewidth',2.0)
136
137 xlabel('Number of Vocoder Filters')
138 ylabel('FT NSIM-Neural Metric Identification')
139 ylim([0 140])
140
141 set(gca,'XTick',filter_n)
142 set(gca,'xscale','log')
143 grid on
144 legend(casestrs([1 2 3 4 5]),'location','SouthWest')
145 %% NSIM MR only
```

```
146 nsim_mr_res=[100*nsim1mean_mr;100*nsim2mean_mr;100*nsim3mean_mr;100*
      nsim4mean_mr;100*nsim5mean_mr];
147 nsim_mr_mdl = fitlm(nsim_mr_res,perceptual_rau)
148 % nsim_mr_mdl_b0=table2array(nsim_mr_mdl.Coefficients(1,1));
149 % nsim_mr_mdl_b1=table2array(nsim_mr_mdl.Coefficients(2,1));
150 % nsim_mr_mdl_x=nsim_mr_mdl_b0+nsim_mr_mdl_b1*nsim_mr_res;
151 nsim_mr_R2 = nsim_mr_mdl.Rsquared.Adjusted;
152 nsim_mr_pValue = nsim_mr_mdl.Coefficients.pValue(2);
153
154
155 figure
156 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
157     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
158     'Speech-TFS + WGN-ENV'};
159 scatter(nsim_mr_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled",'r'
      )
160 hold on
161 scatter(nsim_mr_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7),"
      filled",'b')
162 hold on
163 scatter(nsim_mr_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7),"
      filled",'m')
164 hold on
165 scatter(nsim_mr_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7),"
      filled",'g')
166 hold on
167 scatter(nsim_mr_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7),"
      filled",'c')
168 xlabel('MR NSIM-Neural Metric Identification')
```

```
ylabel('Perceptual Identification')
hold on
plot(1:140,'k')
xlim([0 140])
ylim([0 140])
grid off
legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
text(5, 100, ...
     sprintf('Adj. R^2 = %.3f (p-value <0.001)', nsim_mr_R2), ...
     'FontSize', 10);

figure
casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
   'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
   'Speech-TFS + WGN-ENV'};
filter_n = [1 2 3 6 8 16 32];
plot(filter_n,nsim_mr_mdl.Fitted(1:1*7),'r','linewidth',2.0)
hold on
plot(filter_n,nsim_mr_mdl.Fitted(1*7+1:2*7),'b','linewidth',2.0)
hold on
plot(filter_n,nsim_mr_mdl.Fitted(2*7+1:3*7),'m','linewidth',2.0)
hold on
plot(filter_n,nsim_mr_mdl.Fitted(3*7+1:4*7),'g','linewidth',2.0)
hold on
plot(filter_n,nsim_mr_mdl.Fitted(4*7+1:5*7),'c','linewidth',2.0)

xlabel('Number of Vocoder Filters')
ylabel('MR NSIM-Neural Metric Identification')
ylim([0 140])
```

```
198
199 set(gca,'XTick',filter_n)
200 set(gca,'xscale','log')
201 grid on
202 legend(casestrs([1 2 3 4 5]),'location','NorthWest')
203
204 %% NSIM FT+MR
205 nsim_ft_res=[100*nsim1mean_ft;100*nsim2mean_ft;100*nsim3mean_ft;100*
      nsim4mean_ft;100*nsim5mean_ft];
206 nsim_mr_res=[100*nsim1mean_mr;100*nsim2mean_mr;100*nsim3mean_mr;100*
      nsim4mean_mr;100*nsim5mean_mr];
207 nsim=table(nsim_ft_res,nsim_mr_res,perceptual_rau);
208 nsim_mdl=fitlm(nsim,'interactions','ResponseVar','perceptual_rau'
      ,...
209     'PredictorVars',{'nsim_ft_res','nsim_mr_res'})
210 nsim_R2 = nsim_mdl.Rsquared.Adjusted;
211
212 figure
213 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
214    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
215    'Speech-TFS + WGN-ENV'};
216 scatter(nsim_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled",'r')
217 hold on
218 scatter(nsim_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7),"filled
      ",'b')
219 hold on
220 scatter(nsim_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7),"filled
      ",'m')
221 hold on
```

```
scatter(nsim_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7),"filled
    ",'g')
hold on
scatter(nsim_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7),"filled
    ",'c')
xlabel('NSIM-Neural Metric Identification')
ylabel('Perceptual Identification')
hold on
plot(1:140,'k')
xlim([0 140])
ylim([0 140])
grid off
legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
text(5, 100, ...
     sprintf('Adj. R^2 = %.3f (p-value <0.001)', nsim_R2), ...
     'FontSize', 10);

figure
casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
   'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
   'Speech-TFS + WGN-ENV'};
filter_n = [1 2 3 6 8 16 32];
plot(filter_n,nsim_mdl.Fitted(1:1*7),'r','linewidth',2.0)
hold on
plot(filter_n,nsim_mdl.Fitted(1*7+1:2*7),'b','linewidth',2.0)
hold on
plot(filter_n,nsim_mdl.Fitted(2*7+1:3*7),'m','linewidth',2.0)
hold on
plot(filter_n,nsim_mdl.Fitted(3*7+1:4*7),'g','linewidth',2.0)
```

```
249 hold on
250 plot(filter_n,nsim_mdl.Fitted(4*7+1:5*7),'c','linewidth',2.0)
251
252 xlabel('Number of Vocoder Filters')
253 ylabel('NSIM-Neural Metric Identification')
254 ylim([0 140])
255
256 set(gca,'XTick',filter_n)
257 set(gca,'xscale','log')
258 grid on
259 legend(casestrs([1 2 3 4 5]),'location','NorthWest')
260 %% STMI+NSIM FT
261 nsim_ft_res=[100*nsim1mean_ft;100*nsim2mean_ft;100*nsim3mean_ft;100*
       nsim4mean_ft;100*nsim5mean_ft];
262 stmi_res=[100*stmi1mean;100*stmi2mean;100*stmi3mean;100*stmi4mean
       ;100*stmi5mean];
263 stmi_nsim_ft=table(stmi_res,nsim_ft_res,perceptual_rau);
264 stmi_nsim_ft_mdl=fitlm(stmi_nsim_ft,'interactions','ResponseVar','
       perceptual_rau',...
265     'PredictorVars',{'stmi_res','nsim_ft_res'})
266 stmi_nsim_ft_R2 = stmi_nsim_ft_mdl.Rsquared.Adjusted;
267
268
269
270 figure
271 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
272     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
273     'Speech-TFS + WGN-ENV'};
```

```
274 scatter(stmi_nsim_ft_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled
      ",'r')
275 hold on
276 scatter(stmi_nsim_ft_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7)
      ,"filled",'b')
277 hold on
278 scatter(stmi_nsim_ft_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7)
      ,"filled",'m')
279 hold on
280 scatter(stmi_nsim_ft_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7)
      ,"filled",'g')
281 hold on
282 scatter(stmi_nsim_ft_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7)
      ,"filled",'c')
283 xlabel('STMI+FT NSIM-Neural Metric Identification')
284 ylabel('Perceptual Identification')
285 hold on
286 plot(1:140,'k')
287 xlim([0 140])
288 ylim([0 140])
289 grid off
290 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
291 text(5, 100, ...
292       sprintf('Adj. R^2 = %.3f (p-value <0.001)', stmi_nsim_ft_R2),
      ...
293       'FontSize', 10);
294
295
296 figure
```

```
297
298 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
299     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
300     'Speech-TFS + WGN-ENV'};
301 filter_n = [1 2 3 6 8 16 32];
302 plot(filter_n,stmi_nsim_ft_mdl.Fitted(1:1*7),'r','linewidth',2.0)
303 hold on
304 plot(filter_n,stmi_nsim_ft_mdl.Fitted(1*7+1:2*7),'b','linewidth'
        ,2.0)
305 hold on
306 plot(filter_n,stmi_nsim_ft_mdl.Fitted(2*7+1:3*7),'m','linewidth'
        ,2.0)
307 hold on
308 plot(filter_n,stmi_nsim_ft_mdl.Fitted(3*7+1:4*7),'g','linewidth'
        ,2.0)
309 hold on
310 plot(filter_n,stmi_nsim_ft_mdl.Fitted(4*7+1:5*7),'c','linewidth'
        ,2.0)
311
312 xlabel('Number of Vocoder Filters')
313 ylabel('STMI+FT NSIM-Neural Metric Identification')
314 ylim([0 140])
315 set(gca,'XTick',filter_n)
316 set(gca,'xscale','log')
317 grid on
318 legend(casestrs([1 2 3 4 5]),'location','SouthEast')
319
320
321 % without interaction
```

```
322 stmi_nsim_ft_mdl2=fitlm(stmi_nsim_ft,'perceptual_rau~1+stmi_res+
      nsim_ft_res')
323 stmi_nsim_ft2_R2 = stmi_nsim_ft_mdl2.Rsquared.Adjusted;
324
325 figure
326 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
327    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
328    'Speech-TFS + WGN-ENV'};
329 scatter(stmi_nsim_ft_mdl2.Fitted(1:1*7),perceptual_rau(1:1*7),"
      filled",'r')
330 hold on
331 scatter(stmi_nsim_ft_mdl2.Fitted(1*7+1:2*7),perceptual_rau
      (1*7+1:2*7),"filled",'b')
332 hold on
333 scatter(stmi_nsim_ft_mdl2.Fitted(2*7+1:3*7),perceptual_rau
      (2*7+1:3*7),"filled",'m')
334 hold on
335 scatter(stmi_nsim_ft_mdl2.Fitted(3*7+1:4*7),perceptual_rau
      (3*7+1:4*7),"filled",'g')
336 hold on
337 scatter(stmi_nsim_ft_mdl2.Fitted(4*7+1:5*7),perceptual_rau
      (4*7+1:5*7),"filled",'c')
338 xlabel('STMI+FT NSIM-Neural Metric Identification')
339 ylabel('Perceptual Identification')
340 hold on
341 plot(1:140,'k')
342 xlim([0 140])
343 ylim([0 140])
344 grid off
```

```
345 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
346 text(5, 100, ...
347       sprintf('Adj. R^2 = %.3f (p-value <0.001)', stmi_nsim_ft2_R2),
    ...
348       'FontSize', 10);
349
350 figure
351
352 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
353    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
354    'Speech-TFS + WGN-ENV'};
355 filter_n = [1 2 3 6 8 16 32];
356 plot(filter_n,stmi_nsim_ft_mdl2.Fitted(1:1*7),'r','linewidth',2.0)
357 hold on
358 plot(filter_n,stmi_nsim_ft_mdl2.Fitted(1*7+1:2*7),'b','linewidth'
      ,2.0)
359 hold on
360 plot(filter_n,stmi_nsim_ft_mdl2.Fitted(2*7+1:3*7),'m','linewidth'
      ,2.0)
361 hold on
362 plot(filter_n,stmi_nsim_ft_mdl2.Fitted(3*7+1:4*7),'g','linewidth'
      ,2.0)
363 hold on
364 plot(filter_n,stmi_nsim_ft_mdl2.Fitted(4*7+1:5*7),'c','linewidth'
      ,2.0)
365
366 xlabel('Number of Vocoder Filters')
367 ylabel('STMI+FT NSIM-Neural Metric Identification')
368 ylim([0 140])
```

```
369 set(gca,'XTick',filter_n)
370 set(gca,'xscale','log')
371 grid on
372 legend(casestrs([1 2 3 4 5]),'location','SouthEast')
373 %% STMI+NSIM MR
374 nsim_mr_res=[100*nsim1mean_mr;100*nsim2mean_mr;100*nsim3mean_mr;100*
    nsim4mean_mr;100*nsim5mean_mr];
375 stmi_res=[100*stmi1mean;100*stmi2mean;100*stmi3mean;100*stmi4mean
    ;100*stmi5mean];
376 stmi_nsim_mr=table(stmi_res,nsim_mr_res,perceptual_rau);
377 stmi_nsim_mr_mdl=fitlm(stmi_nsim_mr,'interactions','ResponseVar','
    perceptual_rau',...
378     'PredictorVars',{'stmi_res','nsim_mr_res'})
379 % stmi_nsim_mr_mdl=fitlm(stmi_nsim_mr,'perceptual_rau~1+stmi_res+
    nsim_mr_res') % no interaction
380 stmi_nsim_mr_R2 = stmi_nsim_mr_mdl.Rsquared.Adjusted;
381
382
383 figure
384 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
385    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
386    'Speech-TFS + WGN-ENV'};
387 scatter(stmi_nsim_mr_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled
    ",'r')
388 hold on
389 scatter(stmi_nsim_mr_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7)
    ,"filled",'b')
390 hold on
```

```
391 scatter(stmi_nsim_mr_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7)
      ,"filled",'m')
392 hold on
393 scatter(stmi_nsim_mr_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7)
      ,"filled",'g')
394 hold on
395 scatter(stmi_nsim_mr_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7)
      ,"filled",'c')
396 xlabel('STMI+MR NSIM-Neural Metric Identification')
397 ylabel('Perceptual Identification')
398 hold on
399 plot(1:140,'k')
400 xlim([0 140])
401 ylim([0 140])
402 grid off
403 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
404 text(5, 100, ...
405       sprintf('Adj. R^2 = %.3f (p-value <0.001)', stmi_nsim_mr_R2),
      ...
406       'FontSize', 10);
407
408
409 figure
410 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
411    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
412    'Speech-TFS + WGN-ENV'};
413 filter_n = [1 2 3 6 8 16 32];
414 plot(filter_n,stmi_nsim_mr_mdl.Fitted(1:1*7),'r','linewidth',2.0)
415 hold on
```

```
416 plot(filter_n,stmi_nsim_mr_mdl.Fitted(1*7+1:2*7),'b','linewidth'
      ,2.0)
417 hold on
418 plot(filter_n,stmi_nsim_mr_mdl.Fitted(2*7+1:3*7),'m','linewidth'
      ,2.0)
419 hold on
420 plot(filter_n,stmi_nsim_mr_mdl.Fitted(3*7+1:4*7),'g','linewidth'
      ,2.0)
421 hold on
422 plot(filter_n,stmi_nsim_mr_mdl.Fitted(4*7+1:5*7),'c','linewidth'
      ,2.0)
423
424 xlabel('Number of Vocoder Filters')
425 ylabel('STMI+MR NSIM-Neural Metric Identification')
426 ylim([0 140])
427 set(gca,'XTick',filter_n)
428 set(gca,'xscale','log')
429 grid on
430 legend(casestrs([1 2 3 4 5]),'location','SouthEast')
431
432 %% CCC ENV only
433 CrossCorr_env_res=[100*CrossCorr1_envavg;100*CrossCorr2_envavg;100*
      CrossCorr3_envavg;100*CrossCorr4_envavg;100*CrossCorr5_envavg];
434 ccc_env_mdl = fitlm(CrossCorr_env_res,perceptual_rau)
435 ccc_env_R2 = ccc_env_mdl.Rsquared.Adjusted;
436 ccc_env_pValue = ccc_env_mdl.Coefficients.pValue(2);
437
438 figure
439 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
```

```
440     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
441     'Speech-TFS + WGN-ENV'};
442 scatter(ccc_env_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled",'r'
        )
443 hold on
444 scatter(ccc_env_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7),"
        filled",'b')
445 hold on
446 scatter(ccc_env_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7),"
        filled",'m')
447 hold on
448 scatter(ccc_env_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7),"
        filled",'g')
449 hold on
450 scatter(ccc_env_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7),"
        filled",'c')
451 xlabel('CCC ENV-Neural Metric Identification')
452 ylabel('Perceptual Identification')
453 hold on
454 plot(1:140,'k')
455 xlim([0 140])
456 ylim([0 140])
457 grid off
458 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
459 text(5, 100, ...
460       sprintf('Adj. R^2 = %.3f (p-value = %.3f)', ccc_env_R2,
        ccc_env_pValue), ...
461       'FontSize', 10);
462
```

```
463
464 figure
465 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
466     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
467     'Speech-TFS + WGN-ENV'};
468 filter_n = [1 2 3 6 8 16 32];
469 plot(filter_n,ccc_env_mdl.Fitted(1:1*7),'r','linewidth',2.0)
470 hold on
471 plot(filter_n,ccc_env_mdl.Fitted(1*7+1:2*7),'b','linewidth',2.0)
472 hold on
473 plot(filter_n,ccc_env_mdl.Fitted(2*7+1:3*7),'m','linewidth',2.0)
474 hold on
475 plot(filter_n,ccc_env_mdl.Fitted(3*7+1:4*7),'g','linewidth',2.0)
476 hold on
477 plot(filter_n,ccc_env_mdl.Fitted(4*7+1:5*7),'c','linewidth',2.0)
478 ylim([0 140])
479 xlabel('Number of Vocoder Filters')
480 ylabel('CCC ENV-Neural Metric Identification')
481 set(gca,'XTick',filter_n)
482 set(gca,'xscale','log')
483 grid on
484 legend(casestrs([1 2 3 4 5]),'location','NorthWest')
485
486 %% CCC TFS only
487 CrossCorr_tfs_res=[100*CrossCorr1_tfsavg;100*CrossCorr2_tfsavg;100*
      CrossCorr3_tfsavg;100*CrossCorr4_tfsavg;100*CrossCorr5_tfsavg];
488 ccc_tfs_mdl = fitlm(CrossCorr_tfs_res,perceptual_rau)
489 ccc_tfs_R2 = ccc_tfs_mdl.Rsquared.Adjusted;
490 ccc_tfs_pValue = ccc_tfs_mdl.Coefficients.pValue(2);
```

```
491
492 figure
493 scatter(ccc_tfs_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled",'r'
    )
494 hold on
495 scatter(ccc_tfs_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7),"
    filled",'b')
496 hold on
497 scatter(ccc_tfs_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7),"
    filled",'m')
498 hold on
499 scatter(ccc_tfs_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7),"
    filled",'g')
500 hold on
501 scatter(ccc_tfs_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7),"
    filled",'c')
502 xlabel('CCC TFS-Neural Metric Identification')
503 ylabel('Perceptual Identification')
504 hold on
505 plot(1:140,'k')
506 xlim([0 140])
507 ylim([0 140])
508 grid off
509 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
510 text(5, 100, ...
511      sprintf('Adj. R^2 = %.3f (p-value = %.3f)', ccc_tfs_R2,
    ccc_tfs_pValue), ...
512      'FontSize', 10);
513
```

```
514
515 figure
516 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
517     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
518     'Speech-TFS + WGN-ENV'};
519 filter_n = [1 2 3 6 8 16 32];
520 plot(filter_n,ccc_tfs_mdl.Fitted(1:1*7),'r','linewidth',2.0)
521 hold on
522 plot(filter_n,ccc_tfs_mdl.Fitted(1*7+1:2*7),'b','linewidth',2.0)
523 hold on
524 plot(filter_n,ccc_tfs_mdl.Fitted(2*7+1:3*7),'m','linewidth',2.0)
525 hold on
526 plot(filter_n,ccc_tfs_mdl.Fitted(3*7+1:4*7),'g','linewidth',2.0)
527 hold on
528 plot(filter_n,ccc_tfs_mdl.Fitted(4*7+1:5*7),'c','linewidth',2.0)
529
530 xlabel('Number of Vocoder Filters')
531 ylabel('CCC TFS-Neural Metric Identification')
532 ylim([0 140])
533 set(gca,'XTick',filter_n)
534 set(gca,'xscale','log')
535 grid on
536 legend(casestrs([1 2 3 4 5]),'location','NorthWest')
537
538 %% CCC ENV+TFS
539 CrossCorr_env_res=[100*CrossCorr1_envavg;100*CrossCorr2_envavg;100*
      CrossCorr3_envavg;100*CrossCorr4_envavg;100*CrossCorr5_envavg];
540 CrossCorr_tfs_res=[100*CrossCorr1_tfsavg;100*CrossCorr2_tfsavg;100*
      CrossCorr3_tfsavg;100*CrossCorr4_tfsavg;100*CrossCorr5_tfsavg];
```

```
541 CrossCorr_res=table(CrossCorr_env_res,CrossCorr_tfs_res,
        perceptual_rau);
542 ccc_mdl=fitlm(CrossCorr_res,'interactions','ResponseVar','
        perceptual_rau',...
543     'PredictorVars',{'CrossCorr_env_res','CrossCorr_tfs_res'})
544 ccc_R2 = ccc_mdl.Rsquared.Adjusted;
545
546
547 figure
548 scatter(ccc_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled",'r')
549 hold on
550 scatter(ccc_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7),"filled
        ",'b')
551 hold on
552 scatter(ccc_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7),"filled
        ",'m')
553 hold on
554 scatter(ccc_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7),"filled
        ",'g')
555 hold on
556 scatter(ccc_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7),"filled
        ",'c')
557 xlabel('CCC-Neural Metric Identification')
558 ylabel('Perceptual Identification')
559 hold on
560 plot(1:140,'k')
561 xlim([0 140])
562 ylim([0 140])
563 grid off
```

```
564 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
565 text(5, 100, ...
566       sprintf('Adj. R^2 = %.3f (p-value <0.001)', ccc_R2), ...
567       'FontSize', 10);
568
569
570 figure
571 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
572    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
573    'Speech-TFS + WGN-ENV'};
574 filter_n = [1 2 3 6 8 16 32];
575 plot(filter_n,ccc_mdl.Fitted(1:1*7),'r','linewidth',2.0)
576 hold on
577 plot(filter_n,ccc_mdl.Fitted(1*7+1:2*7),'b','linewidth',2.0)
578 hold on
579 plot(filter_n,ccc_mdl.Fitted(2*7+1:3*7),'m','linewidth',2.0)
580 hold on
581 plot(filter_n,ccc_mdl.Fitted(3*7+1:4*7),'g','linewidth',2.0)
582 hold on
583 plot(filter_n,ccc_mdl.Fitted(4*7+1:5*7),'c','linewidth',2.0)
584
585 xlabel('Number of Vocoder Filters')
586 ylabel('CCC-Neural Metric Identification')
587 ylim([0 140])
588 set(gca,'XTick',filter_n)
589 set(gca,'xscale','log')
590 grid on
591 legend(casestrs([1 2 3 4 5]),'location','NorthWest')
592 %% STMI+CCC ENV
```

```
593 CrossCorr_env_res=[100*CrossCorr1_envavg;100*CrossCorr2_envavg;100*
      CrossCorr3_envavg;100*CrossCorr4_envavg;100*CrossCorr5_envavg];
594 stmi_res=[100*stmi1mean;100*stmi2mean;100*stmi3mean;100*stmi4mean
      ;100*stmi5mean];
595 stmi_ccc_env=table(stmi_res,CrossCorr_env_res,perceptual_rau);
596 stmi_ccc_env_mdl=fitlm(stmi_ccc_env,'interactions','ResponseVar','
      perceptual_rau',...
597     'PredictorVars',{'stmi_res','CrossCorr_env_res'})
598 % stmi_ccc_env_mdl=fitlm(stmi_ccc_env,'perceptual_rau~1+stmi_res+
      CrossCorr_env_res') % no interaction
599 stmi_ccc_env_R2 = stmi_ccc_env_mdl.Rsquared.Adjusted;
600
601 figure
602 scatter(stmi_ccc_env_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled
      ",'r')
603 hold on
604 scatter(stmi_ccc_env_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7)
      ,"filled",'b')
605 hold on
606 scatter(stmi_ccc_env_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7)
      ,"filled",'m')
607 hold on
608 scatter(stmi_ccc_env_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7)
      ,"filled",'g')
609 hold on
610 scatter(stmi_ccc_env_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7)
      ,"filled",'c')
611 xlabel('STMI+CCC ENV-Neural Metric Identification')
612 ylabel('Perceptual Identification')
```

```
613 hold on
614 plot(1:140,'k')
615 xlim([0 140])
616 ylim([0 140])
617 grid off
618 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
619 text(5, 100, ...
620       sprintf('Adj. R^2 = %.3f (p-value <0.001)', stmi_ccc_env_R2),
      ...
621       'FontSize', 10);
622
623
624 figure
625 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
626    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
627    'Speech-TFS + WGN-ENV'};
628 filter_n = [1 2 3 6 8 16 32];
629 plot(filter_n,stmi_ccc_env_mdl.Fitted(1:1*7),'r','linewidth',2.0)
630 hold on
631 plot(filter_n,stmi_ccc_env_mdl.Fitted(1*7+1:2*7),'b','linewidth'
      ,2.0)
632 hold on
633 plot(filter_n,stmi_ccc_env_mdl.Fitted(2*7+1:3*7),'m','linewidth'
      ,2.0)
634 hold on
635 plot(filter_n,stmi_ccc_env_mdl.Fitted(3*7+1:4*7),'g','linewidth'
      ,2.0)
636 hold on
```

```
637 plot(filter_n,stmi_ccc_env_mdl.Fitted(4*7+1:5*7),'c','linewidth'
        ,2.0)
638
639 xlabel('Number of Vocoder Filters')
640 ylabel('STMI+CCC ENV-Neural Metric Identification')
641 ylim([0 140])
642 set(gca,'XTick',filter_n)
643 set(gca,'xscale','log')
644 grid on
645 legend(casestrs([1 2 3 4 5]),'location','SouthWest')
646 %% STMI+CCC TFS
647 CrossCorr_tfs_res=[100*CrossCorr1_tfsavg;100*CrossCorr2_tfsavg;100*
        CrossCorr3_tfsavg;100*CrossCorr4_tfsavg;100*CrossCorr5_tfsavg];
648 stmi_res=[100*stmi1mean;100*stmi2mean;100*stmi3mean;100*stmi4mean
        ;100*stmi5mean];
649 stmi_ccc_tfs=table(stmi_res,CrossCorr_tfs_res,perceptual_rau);
650 stmi_ccc_tfs_mdl=fitlm(stmi_ccc_tfs,'interactions','ResponseVar','
        perceptual_rau',...
651     'PredictorVars',{'stmi_res','CrossCorr_tfs_res'})
652 stmi_ccc_tfs_R2 = stmi_ccc_tfs_mdl.Rsquared.Adjusted;
653
654 figure
655 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
656     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
657     'Speech-TFS + WGN-ENV'};
658 scatter(stmi_ccc_tfs_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"filled
        ",'r')
659 hold on
```

```
660 scatter(stmi_ccc_tfs_mdl.Fitted(1*7+1:2*7),perceptual_rau(1*7+1:2*7)
      ,"filled",'b')
661 hold on
662 scatter(stmi_ccc_tfs_mdl.Fitted(2*7+1:3*7),perceptual_rau(2*7+1:3*7)
      ,"filled",'m')
663 hold on
664 scatter(stmi_ccc_tfs_mdl.Fitted(3*7+1:4*7),perceptual_rau(3*7+1:4*7)
      ,"filled",'g')
665 hold on
666 scatter(stmi_ccc_tfs_mdl.Fitted(4*7+1:5*7),perceptual_rau(4*7+1:5*7)
      ,"filled",'c')
667 xlabel('STMI+CCC TFS-Neural Metric Identification')
668 ylabel('Perceptual Identification')
669 hold on
670 plot(1:140,'k')
671 xlim([0 140])
672 ylim([0 140])
673 grid off
674 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
675 text(5, 100, ...
676      sprintf('Adj. R^2 = %.3f (p-value <0.001)', stmi_ccc_tfs_R2),
      ...
677      'FontSize', 10);
678
679
680 figure
681 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
682    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
683    'Speech-TFS + WGN-ENV'};
```

```
684 filter_n = [1 2 3 6 8 16 32];
685 plot(filter_n,stmi_ccc_tfs_mdl.Fitted(1:1*7),'r','linewidth',2.0)
686 hold on
687 plot(filter_n,stmi_ccc_tfs_mdl.Fitted(1*7+1:2*7),'b','linewidth'
        ,2.0)
688 hold on
689 plot(filter_n,stmi_ccc_tfs_mdl.Fitted(2*7+1:3*7),'m','linewidth'
        ,2.0)
690 hold on
691 plot(filter_n,stmi_ccc_tfs_mdl.Fitted(3*7+1:4*7),'g','linewidth'
        ,2.0)
692 hold on
693 plot(filter_n,stmi_ccc_tfs_mdl.Fitted(4*7+1:5*7),'c','linewidth'
        ,2.0)
694
695 xlabel('Number of Vocoder Filters')
696 ylabel('STMI+CCC TFS-Neural Metric Identification')
697 ylim([0 140])
698 set(gca,'XTick',filter_n)
699 set(gca,'xscale','log')
700 grid on
701 legend(casestrs([1 2 3 4 5]),'location','SouthEast')
702
703 % without interaction
704 stmi_ccc_tfs_mdl2=fitlm(stmi_ccc_tfs,'perceptual_rau~1+stmi_res+
        CrossCorr_tfs_res')
705 stmi_ccc_tfs2_R2 = stmi_ccc_tfs_mdl2.Rsquared.Adjusted;
706
707
```

```
708 figure
709 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
710     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
711     'Speech-TFS + WGN-ENV'};
712
713 scatter(stmi_ccc_tfs_mdl2.Fitted(1:1*7),perceptual_rau(1:1*7),"
        filled",'r')
714 hold on
715 scatter(stmi_ccc_tfs_mdl2.Fitted(1*7+1:2*7),perceptual_rau
        (1*7+1:2*7),"filled",'b')
716 hold on
717 scatter(stmi_ccc_tfs_mdl2.Fitted(2*7+1:3*7),perceptual_rau
        (2*7+1:3*7),"filled",'m')
718 hold on
719 scatter(stmi_ccc_tfs_mdl2.Fitted(3*7+1:4*7),perceptual_rau
        (3*7+1:4*7),"filled",'g')
720 hold on
721 scatter(stmi_ccc_tfs_mdl2.Fitted(4*7+1:5*7),perceptual_rau
        (4*7+1:5*7),"filled",'c')
722 xlabel('Number of Vocoder Filters')
723 ylabel('STMI+CCC TFS-Neural Metric Identification')
724 hold on
725 plot(1:140,'k')
726 xlim([0 140])
727 ylim([0 140])
728 grid off
729 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
730 text(5, 100, ...
```

```
731       sprintf('Adj. R^2 = %.3f (p-value <0.001)', stmi_ccc_tfs2_R2),
      ...
732       'FontSize', 10);
733
734
735 figure
736 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
737     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
738     'Speech-TFS + WGN-ENV'};
739 filter_n = [1 2 3 6 8 16 32];
740 plot(filter_n,stmi_ccc_tfs_mdl2.Fitted(1:1*7),'r','linewidth',2.0)
741 hold on
742 plot(filter_n,stmi_ccc_tfs_mdl2.Fitted(1*7+1:2*7),'b','linewidth'
      ,2.0)
743 hold on
744 plot(filter_n,stmi_ccc_tfs_mdl2.Fitted(2*7+1:3*7),'m','linewidth'
      ,2.0)
745 hold on
746 plot(filter_n,stmi_ccc_tfs_mdl2.Fitted(3*7+1:4*7),'g','linewidth'
      ,2.0)
747 hold on
748 plot(filter_n,stmi_ccc_tfs_mdl2.Fitted(4*7+1:5*7),'c','linewidth'
      ,2.0)
749 xlabel('Number of Vocoder Filters')
750 ylabel('STMI+CCC TFS-Neural Metric Identification')
751 ylim([0 140])
752 set(gca,'XTick',filter_n)
753 set(gca,'xscale','log')
754 grid on
```

```
755 legend(casestrs([1 2 3 4 5]),'location','SouthEast')
756 %% MR NSIM+CCC ENV
757 CrossCorr_env_res=[100*CrossCorr1_envavg;100*CrossCorr2_envavg;100*
        CrossCorr3_envavg;100*CrossCorr4_envavg;100*CrossCorr5_envavg];
758 nsim_mr_res=[100*nsim1mean_mr;100*nsim2mean_mr;100*nsim3mean_mr;100*
        nsim4mean_mr;100*nsim5mean_mr];
759 nsim_mr_ccc_env=table(nsim_mr_res,CrossCorr_env_res,perceptual_rau);
760 nsim_mr_ccc_env_mdl=fitlm(nsim_mr_ccc_env,'interactions','
        ResponseVar','perceptual_rau',...
761     'PredictorVars',{'nsim_mr_res','CrossCorr_env_res'})
762 % nsim_mr_ccc_env_mdl=fitlm(nsim_mr_ccc_env,'perceptual_rau~1+
        nsim_mr_res+CrossCorr_env_res') % no interaction
763 nsim_mr_ccc_env_R2 = nsim_mr_ccc_env_mdl.Rsquared.Adjusted;
764
765 figure
766 scatter(nsim_mr_ccc_env_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"
        filled",'r')
767 hold on
768 scatter(nsim_mr_ccc_env_mdl.Fitted(1*7+1:2*7),perceptual_rau
        (1*7+1:2*7),"filled",'b')
769 hold on
770 scatter(nsim_mr_ccc_env_mdl.Fitted(2*7+1:3*7),perceptual_rau
        (2*7+1:3*7),"filled",'m')
771 hold on
772 scatter(nsim_mr_ccc_env_mdl.Fitted(3*7+1:4*7),perceptual_rau
        (3*7+1:4*7),"filled",'g')
773 hold on
774 scatter(nsim_mr_ccc_env_mdl.Fitted(4*7+1:5*7),perceptual_rau
        (4*7+1:5*7),"filled",'c')
```

```
775 xlabel('MR NSIM+CCC ENV-Neural Metric Identification')
776 ylabel('Perceptual Identification')
777 hold on
778 plot(1:140,'k')
779 xlim([0 140])
780 ylim([0 140])
781 grid off
782 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
783 text(5, 100, ...
784       sprintf('Adj. R^2 = %.3f (p-value <0.001)', nsim_mr_ccc_env_R2)
      , ...
785       'FontSize', 10);
786
787
788 figure
789 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
790     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
791     'Speech-TFS + WGN-ENV'};
792 filter_n = [1 2 3 6 8 16 32];
793 plot(filter_n,nsim_mr_ccc_env_mdl.Fitted(1:1*7),'r','linewidth',2.0)
794 hold on
795 plot(filter_n,nsim_mr_ccc_env_mdl.Fitted(1*7+1:2*7),'b','linewidth'
      ,2.0)
796 hold on
797 plot(filter_n,nsim_mr_ccc_env_mdl.Fitted(2*7+1:3*7),'m','linewidth'
      ,2.0)
798 hold on
799 plot(filter_n,nsim_mr_ccc_env_mdl.Fitted(3*7+1:4*7),'g','linewidth'
      ,2.0)
```

```
800 hold on
801 plot(filter_n,nsim_mr_ccc_env_mdl.Fitted(4*7+1:5*7),'c','linewidth'
        ,2.0)
802
803 xlabel('Number of Vocoder Filters')
804 ylabel('MR NSIM+CCC ENV-Neural Metric Identification')
805 ylim([0 140])
806 set(gca,'XTick',filter_n)
807 set(gca,'xscale','log')
808 grid on
809 legend(casestrs([1 2 3 4 5]),'location','SouthWest')
810 %% MR NSIM+CCC TFS
811 CrossCorr_tfs_res=[100*CrossCorr1_tfsavg;100*CrossCorr2_tfsavg;100*
        CrossCorr3_tfsavg;100*CrossCorr4_tfsavg;100*CrossCorr5_tfsavg];
        nsim_mr_res=[100*nsim1mean_mr;100*nsim2mean_mr;100*nsim3mean_mr
        ;100*nsim4mean_mr;100*nsim5mean_mr];
812 nsim_mr_ccc_tfs=table(nsim_mr_res,CrossCorr_tfs_res,perceptual_rau);
813 nsim_mr_ccc_tfs_mdl=fitlm(nsim_mr_ccc_tfs,'interactions','
        ResponseVar','perceptual_rau',...
814      'PredictorVars',{'nsim_mr_res','CrossCorr_tfs_res'})
815 % nsim_mr_ccc_tfs_mdl=fitlm(nsim_mr_ccc_tfs,'perceptual_rau~1+
        nsim_mr_res+CrossCorr_tfs_res') % no interaction
816 nsim_mr_ccc_tfs_R2 = nsim_mr_ccc_tfs_mdl.Rsquared.Adjusted;
817
818 figure
819 scatter(nsim_mr_ccc_tfs_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"
        filled",'r')
820 hold on
```

```
821 scatter(nsim_mr_ccc_tfs_mdl.Fitted(1*7+1:2*7),perceptual_rau
      (1*7+1:2*7),"filled",'b')
822 hold on
823 scatter(nsim_mr_ccc_tfs_mdl.Fitted(2*7+1:3*7),perceptual_rau
      (2*7+1:3*7),"filled",'m')
824 hold on
825 scatter(nsim_mr_ccc_tfs_mdl.Fitted(3*7+1:4*7),perceptual_rau
      (3*7+1:4*7),"filled",'g')
826 hold on
827 scatter(nsim_mr_ccc_tfs_mdl.Fitted(4*7+1:5*7),perceptual_rau
      (4*7+1:5*7),"filled",'c')
828 xlabel('MR NSIM+CCC TFS-Neural Metric Identification')
829 ylabel('Perceptual Identification')
830 hold on
831 plot(1:140,'k')
832 xlim([0 140])
833 ylim([0 140])
834 grid off
835 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
836 text(5, 100, ...
837       sprintf('Adj. R^2 = %.3f (p-value <0.001)', nsim_mr_ccc_tfs_R2)
      , ...
838       'FontSize', 10);
839
840
841 figure
842 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
843     'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
844     'Speech-TFS + WGN-ENV'};
```

```
845 filter_n = [1 2 3 6 8 16 32];
846 plot(filter_n,nsim_mr_ccc_tfs_mdl.Fitted(1:1*7),'r','linewidth',2.0)
847 hold on
848 plot(filter_n,nsim_mr_ccc_tfs_mdl.Fitted(1*7+1:2*7),'b','linewidth'
      ,2.0)
849 hold on
850 plot(filter_n,nsim_mr_ccc_tfs_mdl.Fitted(2*7+1:3*7),'m','linewidth'
      ,2.0)
851 hold on
852 plot(filter_n,nsim_mr_ccc_tfs_mdl.Fitted(3*7+1:4*7),'g','linewidth'
      ,2.0)
853 hold on
854 plot(filter_n,nsim_mr_ccc_tfs_mdl.Fitted(4*7+1:5*7),'c','linewidth'
      ,2.0)
855
856 xlabel('Number of Vocoder Filters')
857 ylabel('MR NSIM+CCC TFS-Neural Metric Identification')
858 ylim([0 140])
859 set(gca,'XTick',filter_n)
860 set(gca,'xscale','log')
861 grid on
862 legend(casestrs([1 2 3 4 5]),'location','SouthWest')
863 %% FT NSIM+CCC ENV
864 CrossCorr_env_res=[100*CrossCorr1_envavg;100*CrossCorr2_envavg;100*
      CrossCorr3_envavg;100*CrossCorr4_envavg;100*CrossCorr5_envavg];
865 nsim_ft_res=[100*nsim1mean_ft;100*nsim2mean_ft;100*nsim3mean_ft;100*
      nsim4mean_ft;100*nsim5mean_ft];
866 nsim_ft_ccc_env=table(nsim_ft_res,CrossCorr_env_res,perceptual_rau);
```

```
867 nsim_ft_ccc_env_mdl=fitlm(nsim_ft_ccc_env,'interactions','
      ResponseVar','perceptual_rau',...
868     'PredictorVars',{'nsim_ft_res','CrossCorr_env_res'})
869 % nsim_ft_ccc_env_mdl=fitlm(nsim_mr_ccc_env,'perceptual_rau~1+
      nsim_ft_res+CrossCorr_env_res') % no interaction
870 nsim_ft_ccc_env_R2 = nsim_ft_ccc_env_mdl.Rsquared.Adjusted;
871
872 figure
873 scatter(nsim_ft_ccc_env_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"
      filled",'r')
874 hold on
875 scatter(nsim_ft_ccc_env_mdl.Fitted(1*7+1:2*7),perceptual_rau
      (1*7+1:2*7),"filled",'b')
876 hold on
877 scatter(nsim_ft_ccc_env_mdl.Fitted(2*7+1:3*7),perceptual_rau
      (2*7+1:3*7),"filled",'m')
878 hold on
879 scatter(nsim_ft_ccc_env_mdl.Fitted(3*7+1:4*7),perceptual_rau
      (3*7+1:4*7),"filled",'g')
880 hold on
881 scatter(nsim_ft_ccc_env_mdl.Fitted(4*7+1:5*7),perceptual_rau
      (4*7+1:5*7),"filled",'c')
882 xlabel('FT NSIM+CCC ENV-Neural Metric Identification')
883 ylabel('Perceptual Identification')
884 hold on
885 plot(1:140,'k')
886 xlim([0 140])
887 ylim([0 140])
888 grid off
```

```
889 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
890 text(5, 100, ...
891       sprintf('Adj. R^2 = %.3f (p-value <0.001)', nsim_ft_ccc_env_R2)
      , ...
892       'FontSize', 10);
893
894
895 figure
896 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
897    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
898    'Speech-TFS + WGN-ENV'};
899 filter_n = [1 2 3 6 8 16 32];
900 plot(filter_n,nsim_ft_ccc_env_mdl.Fitted(1:1*7),'r','linewidth',2.0)
901 hold on
902 plot(filter_n,nsim_ft_ccc_env_mdl.Fitted(1*7+1:2*7),'b','linewidth'
      ,2.0)
903 hold on
904 plot(filter_n,nsim_ft_ccc_env_mdl.Fitted(2*7+1:3*7),'m','linewidth'
      ,2.0)
905 hold on
906 plot(filter_n,nsim_ft_ccc_env_mdl.Fitted(3*7+1:4*7),'g','linewidth'
      ,2.0)
907 hold on
908 plot(filter_n,nsim_ft_ccc_env_mdl.Fitted(4*7+1:5*7),'c','linewidth'
      ,2.0)
909
910 xlabel('Number of Vocoder Filters')
911 ylabel('FT NSIM+CCC ENV-Neural Metric Identification')
912 ylim([0 140])
```

```
913 set(gca,'XTick',filter_n)
914 set(gca,'xscale','log')
915 grid on
916 legend(casestrs([1 2 3 4 5]),'location','SouthWest')
917 %% FT NSIM+CCC TFS
918 CrossCorr_tfs_res=[100*CrossCorr1_tfsavg;100*CrossCorr2_tfsavg;100*
      CrossCorr3_tfsavg;100*CrossCorr4_tfsavg;100*CrossCorr5_tfsavg];
919 nsim_ft_res=[100*nsim1mean_ft;100*nsim2mean_ft;100*nsim3mean_ft;100*
      nsim4mean_ft;100*nsim5mean_ft];
920 nsim_ft_ccc_tfs=table(nsim_ft_res,CrossCorr_tfs_res,perceptual_rau);
921 nsim_ft_ccc_tfs_mdl=fitlm(nsim_ft_ccc_tfs,'interactions','
      ResponseVar','perceptual_rau',...
922     'PredictorVars',{'nsim_ft_res','CrossCorr_tfs_res'})
923 % nsim_ft_ccc_tfs_mdl=fitlm(nsim_mr_ccc_env,'perceptual_rau~1+
      nsim_ft_res+CrossCorr_tfs_res') % no interaction
924 nsim_ft_ccc_tfs_R2 = nsim_ft_ccc_tfs_mdl.Rsquared.Adjusted;
925
926 figure
927 scatter(nsim_ft_ccc_tfs_mdl.Fitted(1:1*7),perceptual_rau(1:1*7),"
      filled",'r')
928 hold on
929 scatter(nsim_ft_ccc_tfs_mdl.Fitted(1*7+1:2*7),perceptual_rau
      (1*7+1:2*7),"filled",'b')
930 hold on
931 scatter(nsim_ft_ccc_tfs_mdl.Fitted(2*7+1:3*7),perceptual_rau
      (2*7+1:3*7),"filled",'m')
932 hold on
933 scatter(nsim_ft_ccc_tfs_mdl.Fitted(3*7+1:4*7),perceptual_rau
      (3*7+1:4*7),"filled",'g')
```

```
934 hold on
935 scatter(nsim_ft_ccc_tfs_mdl.Fitted(4*7+1:5*7),perceptual_rau
      (4*7+1:5*7),"filled",'c')
936 xlabel('FT NSIM+CCC TFS-Neural Metric Identification')
937 ylabel('Perceptual Identification')
938 hold on
939 plot(1:140,'k')
940 xlim([0 140])
941 ylim([0 140])
942 grid off
943 legend(casestrs([1 2 3 4 5]),'location','NorthWest','box','off')
944 text(5, 100, ...
945       sprintf('Adj. R^2 = %.3f (p-value <0.001)', nsim_ft_ccc_tfs_R2)
      , ...
946       'FontSize', 10);
947
948
949 figure
950 casestrs = {'Speech-ENV + MN-TFS','Speech-ENV + WGN-TFS',...
951    'Speech-TFS + MN-ENV','Speech-TFS + Flat-ENV',...
952    'Speech-TFS + WGN-ENV'};
953 filter_n = [1 2 3 6 8 16 32];
954 plot(filter_n,nsim_ft_ccc_tfs_mdl.Fitted(1:1*7),'r','linewidth',2.0)
955 hold on
956 plot(filter_n,nsim_ft_ccc_tfs_mdl.Fitted(1*7+1:2*7),'b','linewidth'
      ,2.0)
957 hold on
958 plot(filter_n,nsim_ft_ccc_tfs_mdl.Fitted(2*7+1:3*7),'m','linewidth'
      ,2.0)
```

```
959 hold on
960 plot(filter_n,nsim_ft_ccc_tfs_mdl.Fitted(3*7+1:4*7),'g','linewidth'
        ,2.0)
961 hold on
962 plot(filter_n,nsim_ft_ccc_tfs_mdl.Fitted(4*7+1:5*7),'c','linewidth'
        ,2.0)
963
964 xlabel('Number of Vocoder Filters')
965 ylabel('FT NSIM+CCC TFS-Neural Metric Identification')
966 ylim([0 140])
967 set(gca,'XTick',filter_n)
968 set(gca,'xscale','log')
969 grid on
970 legend(casestrs([1 2 3 4 5]),'location','SouthWest')
```

Bibliography

Bruce, I. C., Erfani, Y., & Zilany, M. S. (2018). A phenomenological model of the synapse between the inner hair cell and auditory nerve: Implications of limited neurotransmitter release sites. Hearing research, 360, 40-54.

Chi, T., Gao, Y., Guyton, M. C., Ru, P., & Shamma, S. (1999). Spectro-temporal modulation transfer functions and speech intelligibility. The Journal of the Acoustical Society of America, 106(5), 2719-2732.

Elhilali, M., Chi, T., & Shamma, S. A. (2003). A spectro-temporal modulation index (STMI) for assessment of speech intelligibility. Speech communication, 41(2-3), 331-348.

French, N. R., & Steinberg, J. C. (1947). Factors governing the intelligibility of speech sounds. The journal of the Acoustical society of America, 19(1), 90-119.

Heinz, M. G., & Swaminathan, J. (2009). Quantifying envelope and fine-structure coding in auditory nerve responses to chimaeric speech. Journal of the Association for Research in Otolaryngology : JARO, 10(3), 407–423.

Hines, A., & Harte, N. (2012). Speech intelligibility prediction using a neurogram similarity index measure. Speech Communication, 54(2), 306-320.

Jackson, B. S., & Carney, L. H. (2005). The spontaneous-rate histogram of the auditory nerve can be explained by only two or three spontaneous rates and long-range dependence. Journal of the Association for Research in Otolaryngology, 6, 148-159.

Joris, P. X. (2003). Interaural time sensitivity dominated by cochlea-induced envelope patterns. Journal of Neuroscience, 23(15), 6345-6350.

Joris, P. X., Louage, D. H., Cardoen, L., & van der Heijden, M. (2006). Correlation index: a new metric to quantify temporal coding. Hearing research, 216, 19-30.

Louage, D. H., van der Heijden, M., & Joris, P. X. (2004). Temporal properties of responses to broadband noise in the auditory nerve. Journal of neurophysiology, 91(5), 2051-2065.

Rosen, S. (1992). Temporal information in speech: acoustic, auditory and linguistic aspects. Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences, 336(1278), 367-373.

Shamma, S., & Lorenzi, C. (2013). On the balance of envelope and temporal fine structure in the encoding of speech in the early auditory system. The Journal of the Acoustical Society of America, 133(5), 2818-2833.

Smith, Z. M., Delgutte, B., & Oxenham, A. J. (2002). Chimaeric sounds reveal dichotomies in auditory perception. Nature, 416(6876), 87-90.

Swaminathan, J., & Heinz, M. G. (2012). Psychophysiological analyses demonstrate the importance of neural envelope coding for speech perception in noise. Journal of Neuroscience, 32(5), 1747-1756.

Taghavi SMR, Mohammadkhani G, Jalilvand H. Speech Intelligibility Index: A Literature Review. Aud Vestib Res. 2022;31(3):148-57.

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), 600-612.

Wirtzfeld, M. R., Ibrahim, R. A., & Bruce, I. C. (2017). Predictions of Speech Chimaera Intelligibility Using Auditory Nerve Mean-Rate and Spike-Timing Neural Cues. Journal of the Association for Research in Otolaryngology : JARO, 18(5), 687–710.

Zilany, M. S., Bruce, I. C., Nelson, P. C., & Carney, L. H. (2009). A phenomenological model of the synapse between the inner hair cell and auditory nerve: long-term adaptation with power-law dynamics. The Journal of the Acoustical Society of America, 126(5), 2390-2412.