

TEL AVIV UNIVERSITY

THE IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING
The Zandman-Slaner School of Graduate Studies

An Efficient Simulation Tool For the Auditory System By Parallel Processing

A thesis submitted toward the degree of
Master of Science in Electrical and Electronic Engineering

by
Yonatan Koral

This research was carried out in the
Department of Electrical Engineering - Systems,
under the supervision of Prof. Shlomo Weiss and Prof. Miriam
Furst-Yust

October, 2018

Acknowledgments

.....

Abstract

This work present an Auditory emulation program that solves Auditory Neural response and detects Hearing level based on Parallel Time domain non-linear solution of the Cochlea. Previous solution used parallel calculation of Basilar membrane velocity and than calculated serial Neural Response along time and longitudinal dimension, this solution limited has efficiency by both copy large arrays and perform serial computation. using GPU to massive parallel computing JND calculation speed can be accelerated by factor of 80–400, which make this solution valuable for both calculate hearing level for single pitch signal and spoken words, both with variable kinds noise or in silence, and measuring effects of different hearing aid prescriptions on the JND.

Contents

1	Introduction	1
2	Description of The Human Ear Anatomy	4
2.1	Basilar Membrane Velocity Calculation	4
2.2	Model of the Inner hair cell auditory nerve synapse	10
2.2.1	ANR response	14
2.3	Hearing Threshold, JND, Based on Auditory Nerve	15
3	GPU With CUDA Architecture Updates	20
3.1	Difference Between Generations of NVidia architecture	22
3.2	NVidia Nsight	27
4	Cochlear Model Implementation Updates	30
4.1	Single precision computations updates	30
4.1.1	Serial Solution of boundary conditions equations	30
4.1.2	Serial Solution of initial conditions equations	33
4.1.3	Run Time Estimation	35
4.2	Parallelizing the algorithm	36
4.2.1	Parallelism in time dimension	36

4.2.2	Parallelism in longitudinal dimension	38
4.3	Updating The computing algorithm to reduce errors	40
4.3.1	Error Measurements	41
5	Implementing Neural Response and JND evaluation	48
5.1	Neural Response Calculation	48
5.2	JND Calculation	51
5.2.1	Calculate Fisher Information	51
6	Optimizations	57
6.1	Parallel Input Generation	57
6.1.1	Requirements	57
6.1.2	Normalization	60
6.2	Optimizing For Kepler Architecture	61
6.3	Optimizing For Pascal Architecture	64
6.4	Testing Congestion With Nsight	66
6.5	Optimizing Convergence Parameters	66
6.6	Optimize JND calculation for single tones	67
6.7	Optimizing Execution Flow	70
7	Hearing Aids Effectiveness	75
7.1	Characteristics	75
7.2	Hearing Aid History	76
7.3	Hearing Aid, Gain Calculations	77

7.3.1	Hearing Aids Simulation	77
7.3.2	Database Generation	78
7.4	Hearing Aid, OHC damage	79
7.5	Hearing Aid, Combined OHC and IHC damage	89
8	Run time Comparison	96
8.1	Run Times for different configurations	96
8.1.1	Comparing run-time for different tasks	97
8.2	Acceleration from work-flow modifications	100
8.3	Additional Experiment	100
9	Conclusions	104
	Appendices	106
A	Cochlear Constants	107
B	Architecture Features Comparison	109
C	Hardware Used	112

List of Figures

2.1	Detailed description of human ear anatomy from [1]	5
2.2	the structure of Cochlea	5
2.3	Cochlear model of pressures applied	6
2.4	Equivalent electrical circuit of outer hair cell	9
3.1	Comparison of Kepler (CC 3.x) and Maxwell (5.x) SM based on figures from [30] and [32]	26
4.1	Flow chart to replace $t + \Delta t$	31
4.2	Partitioning in the time dimension each bar represents a time interval handled by one CUDA block. Overlapping sections are discarded	37
4.3	Envelope of Basilar Membrane velocity section in response for 4KHz tone 36dB with old(Red) and new(Blue) interpolation methods	42
4.4	Result yielded by new interpolation method for $P_{in}(t)$, at 4KHz the nerves cant follow the membrane phase and gives constant yield which is proportional of $ P_{in}(t) $ as observed experimentally	45
4.5	Result for old method of interpolation, we see nerves response change phase both temporarily and spatially in response for the degraded $d_{precision}$	45

4.6	Error Mean difference show improvement for All frequencies and powers of relative error of $0.4 \cdot 10^{-3}$, for lower frequencies error is lowered for less then half of its former value	46
4.7	Error Mean difference show improvement for All frequencies, the effect however is most significant for 4KHz, error minimized from 16% to less than 1% for 40dB makes the new interpolation fit to calculate JND where the old version will err significantly	47
5.1	Flow chart of the Calculation of Auditory Nerve Response.	49
5.2	Flow chart of the Just Noticeable Difference Calculation.	52
6.1	Flow Chart of Input generation. Continues at Fig. 6.2	58
6.2	Flow Chart of Input generation continues from Fig. 6.1	59
6.3	Example for Eq. (6.2) implementation.Generated Input for $\Delta\alpha_{SL}$ as 40dB and 50dB SPL and α_{NL}^* of 0 (Noiseless), Pure tones at frequencies of 0.5, 1 and 4 K Hz.	62
6.4	Comparison of Kepler Register Per Thread Requirements for implementations M1 - M6 from Section 6.2	65
6.5	Comparison of Pascal Run times for different configurations, each number indicates $\frac{runtime(M1)}{runtime(Configuration)}$. for example, if M1 to 3.52 seconds M2 will take $\frac{3.52}{1.66} = 2 seconds$	65

6.6	Comparison of run time on Nsight as debug mode and on Matlab as run time, Intervals amount tested, M5-8 described at Sections 6.2 and 6.3. Instructions Per cycle can be maximally 8 if there are 4 warps that can issue 2 instructions every cycle. this is not the case here, as stall issues block warps around 75% of the clock cycles. It is also noticeable that Nsight run time is negatively predict tested run time	67
6.7	Testing JND accuracy and algorithm acceleration as function of minimum Δt in Lipschitz criteria. We tested 2 criteria. First run time acceleration factor over base algorithm run time at convergence speed of $10^{-15}cm$. Second is percent of JND measurements differs more than $2dB$ from reference algorithm of Oded and Furst 36 to avoid large differences.	68
6.8	Division of Interval to measure JND, $T_{shortened}$ from Eq. (6.8a) and Int from Eq. (6.8b) and t_{BOP} described at Fig. 4.2	69
6.9	Testing JND accuracy as function of interval length,X axis is tested interval length in seconds, Y axia is normalized covariance from Eq. (6.11e)	70
6.10	Flow chart of program.Input and Cochlear solver	73
6.11	Flow chart of program.Output and release memory	74
7.1	Normal Hearing model Auditory Nerve(ANR) response for the Hebrew word SHEN spoken by female at comfortable sound level,we see differences of detected signals, Simulation results close to real experiments. Detection of consonant SH shown on the neural response graph	80

7.2	Patient with OHC at 25% and IHC at 87% ANR response for the Hebrew word SHEN spoken by female at comfortable sound level, we see differences of detected signals, comparing ANR and JND to healthy mode Fig. 7.1. We see for hearing impaired patient most consonants disappeared, the remain sound is weak HE, make the word difficult if not impossible to understand, also note that with IHC loss is not constant the IHC's pass 2.5-3KHZ frequencies are damaged substantially more. this is also can be seen by the JND SPL graphs, using crowd noise passed through Butter-worth LPF with passband of 800Hz, spectrogram at Figure 7.3, stop-band of 1.2KHZ with attenuations $-3dB$ and $-30dB$ respectively	81
7.3	show normalized energy density of noise for JND test at Figure 7.1(a) and Figure 7.2(a)	82
7.4	show different Audiogram HL for patient with OHC damage, without aid and with different aids prescriptions, described in 7.4 calculated by the program	84
7.5	Compare the 4 hearing aids prescriptions with Fig. 7.6, Pogo is the only one that accentuate the higher frequencies	85
7.6	Compare the 4 hearing aids prescriptions with Fig. 7.5, Berger has much more amplification.	86

7.7	Comparative results of ANR for Hebrew word SHEN, spoken by female at comfortable hearing level, with different aids, response without aid shown at Fig. 7.2(b). both POGO and NAL methods are under severely amplified, the SH consonant response is effectively disappeared, Berger methods shown at Fig. 7.8	87
7.8	Comparative results of ANR for Hebrew word SHEN, spoken by female at comfortable hearing level, with different aids, response without aid shown at Figure 7.2(b). While the Berger methods give better results than NAL and POGO shown at Fig. 7.7, it still significantly different from healthy cochlea. this makes OHC damage unfit for hearing aids to fix.	88
7.9	Model with OHC 40% and IHC 68% ANR when compared to Normal Hearing Patient shown at Fig. 7.1. Response for the Hebrew word SHEN spoken by female at comfortable sound level,compare it to the response of patient with sever OHC damage to the same signal at Fig. 7.2(b), JND degrading is more significant for the lower frequency, as seen both for the SPL audiogram at Fig. 7.9(a) and for the ANR at Fig. 7.9(b), where for this patient very low response for the SH consonant remain, the low frequency component of the E vowel is much weaker than other patient.	90

7.10	The Comparison for patient Audiogram HL with different prescriptions, major differences from Figure 7.10 are that the critical frequencies 2-4KHz are amplified much more than lower. this both prevent low pass noise amplification and avoid upward masking from the signal lower bands components.	91
7.11	Compare the 4 hearing aids prescriptions, Berger methods shown at Fig. 7.12. NAL and POGO significantly under amplify at higher frequencies relatively to berger methods	92
7.12	Compare the 4 hearing aids prescriptions,NAL and POGO shown at Fig. 7.11, Berger has much more amplification and pogo is the only one that accentuate the higher frequencies	93
7.13	Comparative results of ANR for Hebrew word SHEN, spoken by female at comfortable hearing level, with different aids, response without aid shown at Figure 7.9(b). The NAL and POGO methods amplification is insufficient	94
7.14	Comparative results of ANR for Hebrew word SHEN for Berger Behind and Inside the Ear, spoken by female at comfortable hearing level, with different aids, response without aid shown at Figure 7.9(b).This methods gives much better amplification from POGO and NAL shown at Fig. 7.13	95

8.1	Comparison for output copy time, HD and RAM for different buffer sizes, The left Y has the combined time of write be the program and read by Matlab the results. the right Y axis has relation between those times for HD and RAM	101
8.2	Comparison for output copy time, HD and RAM for different buffer sizes, The left Y has the combined time of write be the program and read by Matlab the results. the right Y axis has relation between those times for HD and RAM	102
8.3	test JND for healthy patient as function of signal time. JND for every tested tone measured as function of interval.22 intervals were 100 to 500msec by jump of 100msec. and 1 to 9 seconds by jump of half second. 10 levels of power 10 to 100dB for 7 frequencies. accumulates to 6055 seconds of audio. times shown at 8.1, \mathcal{M}	103

List of Tables

8.1	Comparing run-time for different tasks, with CPU only, GPU + CPU and all heavy calculation tasks on GPU measured in seconds.	99
A.1	Parameters for cochlear equations solution.	108
B.1	NVidia GPUs Technical specifications parameters summarized from [34, Pg. 218-220]	111
B.2	NVidia GPUs Architecture specifications parameters summarized from [34, Pg. 83-84]	111
C.1	Processor configuration for sequential, single-core computation.	112
C.2	Hardware configuration for the parallel code.	113

Abbreviations

AC Alternate Current

AI All Information. Hearing level estimation method that account for neural spikes rate and relative position to each other

AN Auditory Nerve. Neural cells connected to the cochlea, used for detect auditory signal

ANR Auditory Nerve Response. Auditory Nerves level of firing electrical spikes

BM Basilar Membrane. Stiff membrane that sepearate 2 liquid tubes inside the cochlea. it resonates in response for air pressure and response to high frequency most strongly near the oval window and low near the apex.

BMV Basilar Membrane Velocity. Basilar Membrane velocity measured as velocity oscillation per longitudnal section

CHH Children of Hard Hearing. Children that have hearing loss and in need for hearing aid

CNH Children of Normal Hearing. Children that have no hearing loss

CPU Central Processing Unit. Electronic circuit capable of executing instructions from main memory of the computer. compared to the GPU, it has more cache memory per core than GPU and its used as control unit for the computer, compare to the GPU that is mainly used for massively parallel computing programs

CRLB Cramer Raw Lower Bound. Lower boundary on the variance of estimator on fixed but unknown parameter

CUDA Compute Unified Device Architecture. software development kit and application programming interface for parallel computing on CUDA enabled hardware of NVidia

dB Decibels

GPU General Processing Unit. Electronic circuit capable of executing the same instructions on multiple data segments in parallel. Used mainly for physical simulations Machine learning and Deep learning algorithms and graphics

HA Hearing Aid. Electro Mechanical Device that amplifies sound to assist hearing impaired

HL Hearing Loss. difference between hearing levels and normal hearing person per frequency measured in dB

HSR High Spontaneous Rate. Property for group of Auditory nerve that their firing rate is responsive to small changes in BM velocity, they have high firing rate of spikes even in stimulation less environment

IHC Inner Hair Cells. muscle cells structured hair like that connect to the amplifying membranes in the cochlea from one side and auditory nerves on the other side [51]. they trans-duce mechanical vibrations to electrical signal and amplifying it as well

JND Just Noticeable Difference. The signal difference of magnitude over reference level necessary for sensory detection

LSR Low Spontaneous Rate. Property for group of Auditory nerve that their firing rate is responsive only to large changes in BM velocity, they have very low firing rate of spikes in stimulation less environment

MSR Medium Spontaneous Rate. Property for group of Auditory nerve that thier firing rate is responsive to medium changes in BM velocity, they have medium firing rate of spikes in stimulation less enviroment

NAL National Acoustic Laboratories. Australian laboratories eponymous to a method of calculating hearing aid gain per hearing loss at given frequency

NH Normal Hearing. Level of hearing defined that person does not need hearing aid to detect auditory input

NHPP non homogeneous Poission Process. A memoryless process of random occurring events so that inter arrival time is not constant

NL Noise Level. Magnitude of noise that is interfere with signal to be detected by auditory nerves

OHC Outer Hair Cells. muscle cells structured hair like that connect the amplifying membranes in the cochlea [51]. they mechanically amplifying sound pressure

OW Oval Window. A membrane opening that connects the middle ear to the cochlea liquid tube

POGO Prescription of Gain and Output. function that maps hearing loss to hearing aid prescribed gain at frequencies. characterized by half insertion gain + constant depend on frequency

QoL Quality of Life. Quantifying Measurement of human wellness

RMS Rate Mean Square. Hearing level estimation method that account for neural spikes average rate on time section but not their relative position to each other

SFU Special Functional Unit. Hardware implemented complex mathematical functionality on NVidia's GPU. Such as division and logarithms. while this functions are approximations, they are much faster the software implemented alternative for some cases.

SL Signal Level. Magnitude of signal that is desired to be detect by auditory nerves

SPL Sound Pressure Level. Logarithmic scale used to measure air vibration applied force over the ear relate to reference level, measured in dB

SPL Reference Sound Pressure Level Reference - pressure of sound wave at 0 dB

Chapter 1

Introduction

hearing impairment has become a common phenomenon, with significant part of communicating is speech. hearing necessary for both earning and personal communication, while the amount for Quality of Life(QoL) reduced due to hearing loss is undetermined [54]. it is generally agreed that the impairment reduce QoL. the hearing loss (HL) is phenomenon on continuous scale and cut offs barrier to determine hearing impaired is disagreed upon across different researches [38]. by using standardized methods to set unified threshold for hearing impairment. its also the second common disability by measure years of life with the impairment. In 2012 World Health Organization estimated 360 million people (5.3% of world population) have disabling Hearing Impairment.for adults, age 55-74 above 30% has lost more than 30dB in their better Ear [38], however the majority, 80% of the adults need hearing aid do not use them [12]. those who does use has significant HL more than 10 years before getting an aid [9]. examination of the reasons at [13] shows that for 7 out of researches 15-30% patients who does not use HA does not benefit from the device and for 5 researches 22-52% suffer from noisy background. For children, multiple longitudinal researches occurred to measure hearing aid effectiveness, adoption of early screening and inter-

vention ensure that HA and Cochlear Implants adjusted to children at very young age. parallel research on children with normal hearing (CNH) show that amount of vocal input effect language development, researchers assumed that if vocal input present children will uptake the information. for children with Hard Hearing (CHH) this assumption fails due to difficulty of understanding in noisy environment or low power speech leads to variable uptake and reduction of available processed oral communication, degradation varied but includes low pass filtering and spectral reduction that cause final -s to be inaudible and lost of experience for many morphemes [44]. multiple causes for varied uptake include distance to the parent, noise from the environment and HA fitnesses. Some of the studies found that CHH with HA has linguistic development one standard deviation below their age with HA fitting age not predictive [48]. Despite controversy about HA effectiveness for CHH several studies found that audibility assistance increase language understanding for school children [45, 22] but not to level of CNH, other studies does show HA contribute significantly to language understanding [55]. This unclear picture suggests that more precise measurements are needed to test effectiveness of HA and better fitting methods are required. Hearing Aids are adapted for user by examining the patient hearing threshold for pure tones and testing speech discrimination in quiet. the user than get prescription for duration of examination and return to report experience and adjustments, this process can occur several times due to lack of objective method to test prescription effectiveness. Solution of the cochlear model in time domain was done by Oded and Furst 36, however the solution was implemented on CPU, and is very slow.real time solution

requires Massive Parallel Computation. Sabo et al. 39 implemented part of the model on Commodity GPU, but solution was still slow. We implemented the entire model on GPU, including Hearing level inference and effects of varying prescription of Hearing Aids on the model.

The remainder of thesis is composed as follows. Chapter 2 describes the cochlear model simulated by our program. Chapter 3 compares different NVidia architectures by generation and their effect on program performance. Section 6.1 describe parallelization for signal and noise at various power levels. At Chapter 4 we describe improving of [40] algorithm and test performance for multiple execution profiles. In Section 5.1 we describe the calculation of auditory nerve response from Basilar Membrane Velocity. In Section 5.2 we present parallel algorithm to calculate JND directly on GPU from Auditory Nerve Response. Chapter 7 describes program usage to diagnose patient and fit hearing aid prescription. We present results at Chapter 8 and conclusion at Chapter 9

Chapter 2

Description of The Human Ear Anatomy

2.1 Basilar Membrane Velocity Calculation

Our work implements Furst 14 model of the mammalian ear. The mammalian ear is composed of the outer ear, the middle ear, and the inner ear. The outer ear includes the pinna, the ear canal, and the ear drum. The middle ear is an air-filled cavity behind the ear drum, which includes three small ear bones, the ossicles. The inner ear includes a snail-shaped structure, the cochlea (see schematic description in Figure Figure 2.3). The sound is directed by the outer ear through the ear canal to the eardrum. When sound strikes the ear Hearing Loss drum, the movement is transferred through the three bones of the middle ear to a flexible tissue called the oval window, finally reaching the upper fluid-filled ducts of the cochlea (see Fig. 2.3). The upper cochlear ducts are called scala vestibuli, and the bottom duct is referred to as scala tympani. The space between the top and bottom ducts is labeled as scala media.

The middle ears task is to match the impedance of the sound pressure in the air to that of the fluid. Movement of the fluid inside the upper cochlear duct results in a pressure difference between the upper and lower ducts. This pressure difference in

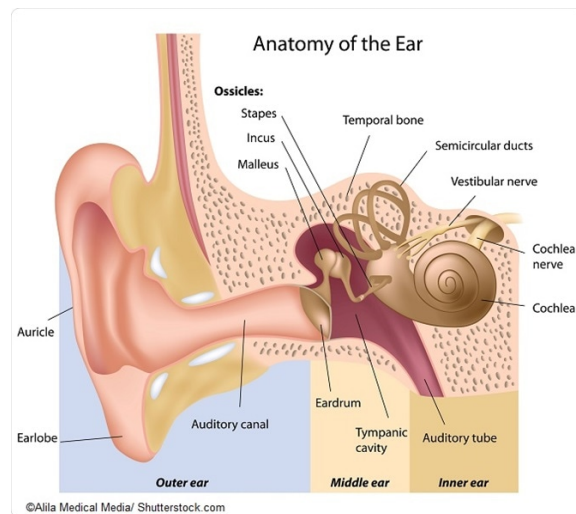


Figure 2.1: Detailed description of human ear anatomy from [1]

turn causes the basilar membrane (the membrane that separates the scala tympani and scala media) to move.

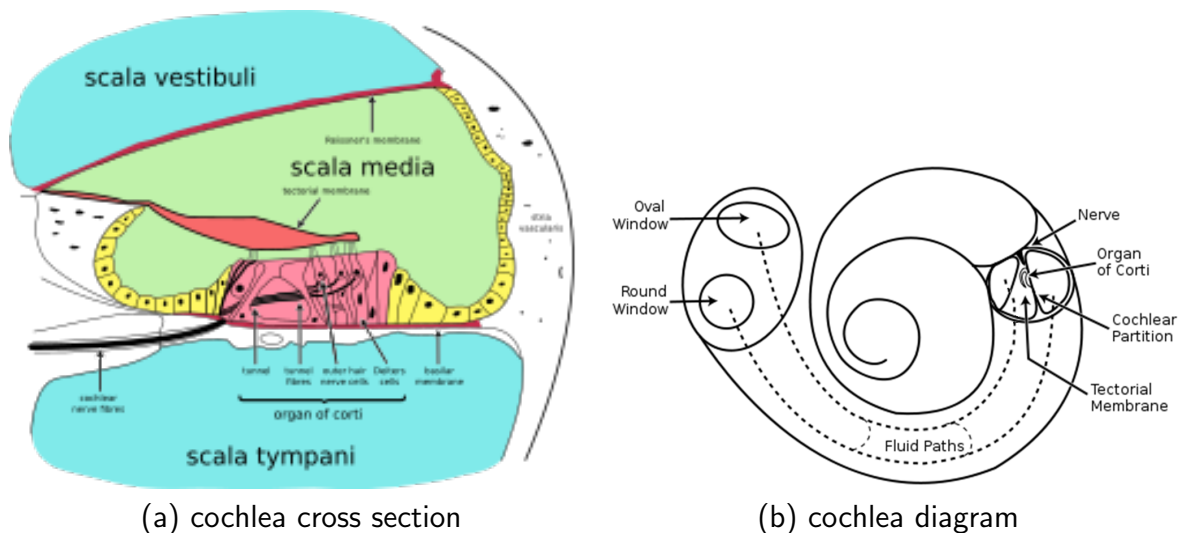


Figure 2.2: the structure of Cochlea

Two types of auditory receptor cells inhabit the scala media, the inner and outer hair cells. The defining feature of those cells is the hair bundle on top of each cell. The hair bundle comprises dozens to hundreds of stereocilia, which are cylindrical actin-filled rods. The stereocilia are immersed in endolymph, a fluid that is rich in potassium and characterized by an endocochlear potential of +80 mV. The stereocilia move with

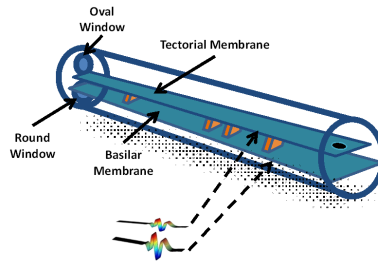


Figure 2.3: Cochlear model of pressures applied

the basilar membrane displacement. Their deflection opens mechanically gated ion channels that allow any small, positively charged ions (primarily potassium and calcium) to enter the cell. The influx of positive ions from the endolymph in the scala media depolarizes the cell, resulting in a receptor potential. The roles of the OHCs and IHCs on the function of the cochlea are very different. While the OHCs act as local amplifiers, the IHCs innervate the auditory nerve. The OHCs lay on the basilar membrane, and their upper part is embedded in a gel-like membrane, the tectorial membrane (TM). An increase in the OHC receptor potential causes a decrease in its length [5], which in turn enhances the BM movement. The hair bundles of the IHC move freely in the scala media. The change in their receptor potential opens voltage-gated calcium channels that release neurotransmitters at the basal end of the cell, which trigger action potentials in the attached nerve.

Modeling the human ear requires a detailed model of the cochlea and the middle and outer ears [6, 39, 21, 58, 4, 15]. A common approach is to model the inner ear as a one-dimensional structure with the cochlea regarded as an uncoiled structure with two fluid-filled compartments with rigid walls that are separated by an elastic partition, the basilar membrane. The cochlear partition, whose mechanical properties are

describable in terms of point-wise mass density, stiffness, and damping, is regarded as a flexible boundary between scala tympani and scala vestibuli. Thus, at every point along the cochlear duct, the pressure difference $P(x, t)$ across the partition drives the partitions velocity. By applying fundamental physical principles, such as the conservation of mass and the dynamics of deformable bodies, the differential equation for P is obtained by [14]

$$\frac{\partial^2 P(x, t)}{\partial x^2} = \frac{2\rho\beta}{A} \frac{\partial^2 \xi_{BM}(x, t)}{\partial t^2} \quad (2.1)$$

where ξ_{BM} is the BM displacement, A represents the cross-sectional area of scala tympani and scala vestibuli, β is the BM width, and ρ is the density of the fluid in both the scala vestibuli and the scala tympani. The pressure on the BM (P_{bm}) is a result of both the difference in fluid pressure and the pressure caused by the OHCs (P_{ohc}). The relation between the pressures of BM, TM, and OHC is shown in [36] Fig. 2.3, which can be interpreted as

$$\left. \begin{aligned} P_{BM}(x, t) &= P(x, t) + P_{OHC}(x, t) \\ 0 &= P_{OHC}(x, t) + P_{tm}(x, t) \end{aligned} \right\} \quad (2.2)$$

The mechanical properties of both BM and TM are simulated as second-order oscillators that yield

$$\left. \begin{aligned} P_{bm} &= M_{bm}(x) \cdot \frac{\partial^2 \xi_{bm}(x, t)}{\partial t^2} + R_{bm}(x) \cdot \frac{\partial \xi_{bm}(x, t)}{\partial t} + K_{bm}(x) \cdot \xi_{bm}(x, t) \\ P_{tm} &= M_{tm}(x) \cdot \frac{\partial^2 \xi_{tm}(x, t)}{\partial t^2} + R_{tm}(x) \cdot \frac{\partial \xi_{tm}(x, t)}{\partial t} + K_{tm}(x) \cdot \xi_{tm}(x, t) \end{aligned} \right\} \quad (2.3)$$

where K_{bm} , K_{tm} , R_{bm} , R_{tm} , M_{bm} , and M_{tm} are the effective stiffness, damping, and mass per unit area of BM and TM, respectively (see Table A.1). The TM displacement

is defined as ξ_{tm} . Since the OHCs lie between the two membranes, their displacement is considered as

$$\xi_{ohc} = \xi_{tm} - \xi_{bm} \quad (2.4)$$

Each OHC is modeled by two sections, the apical and basal parts. The apical part is directed toward the endolymph of the gap between the TM and the reticular lamina (RL), while the basolateral part is embedded in the perilymph next to the supporting cells that are aligned along the BM. When the OHCs stereocilia move due to the relative displacement of the BM and the TM, the conductance of the apical part of the OHC is affected, which in turn causes a flow of potassium and calcium ions to the endolymph. Thus, a voltage drop is developed on the basal part of the OHC membrane. [8]

An outer hair cell model is described by an equivalent electrical circuit in Fig. 2.4 [6, 27]. The apical part is presented by its variable conductance ($G_a \approx \xi_{ohc}$) and its constant capacitance (C_a), while the basal part is presented by its constant conductance and capacitance, G_b and C_b , respectively. The electrical potential of the endolymph is $V_{sm} = 80mV$, and the perilymph resting potential is $\psi_0 = 70mV$. Solving the equivalent electrical circuit by using Kirchhoff laws yields the differential equation for ψ_{ohc} , the OHCs membrane voltage:

$$\frac{d\psi_{OHC}}{dt} + \omega_{OHC} \cdot (\psi - \psi_0) = \eta \cdot \xi_{OHC} \quad (2.5)$$

where $\omega_{ohc} \approx G_b/C_b = 1000Hz$, which represents the cutoff frequency of the OHCs

membrane and $\eta = \alpha V_{sm} / (C_b + C_a) = \text{const.}$ (see Table A.1).

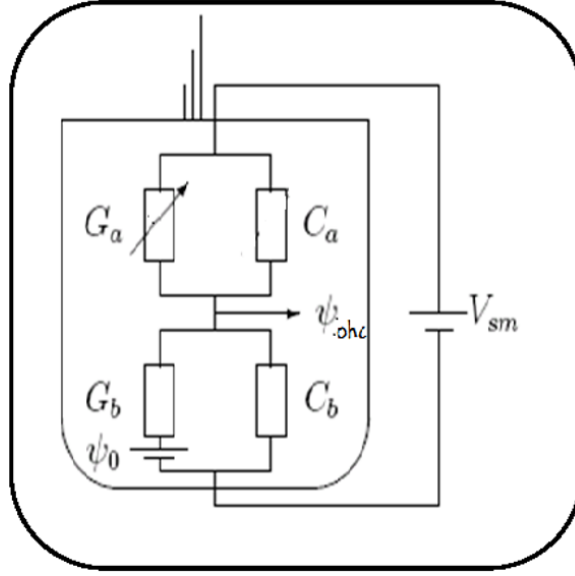


Figure 2.4: Equivalent electrical circuit of outer hair cell

An OHCs length changes due to the electrical potential developed on the OHC membrane and is defined as Δl_{ohc} . It is usually described as a sigmoid function [56, 18, 29]:

$$\Delta l_{ohc} = \alpha_s \frac{e^{-2 \cdot \alpha_l \cdot \psi} - 1}{e^{-2 \cdot \alpha_l \cdot \psi} + 1} = \alpha_s \cdot \tanh(-\alpha_l \cdot \psi) \quad (2.6)$$

where α_l and α_s are constants (see Table A.1).

The pressure developed by each OHC (P_{ohc}) is obtained from the spring properties of the OHC. Lets define $\gamma_{ohc}(x)$ as the OHC effective index. It represents the effective distribution of the OHCs along the cochlear partition. Therefore, the OHC pressure is obtained by

$$P_{ohc}(x, t) = \gamma_{ohc}(x) \cdot K_{ohc}(x) \cdot [\xi_{ohc}(x, t) - \Delta l_{ohc}(x, t)] \quad (2.7)$$

where K_{ohc} is the OHCs stiffness (Table A.1). A cochlea with no active OHC is obtained by $\gamma_{ohc}(x) = 0$, whereas $0.5 \leq \gamma_{ohc}(x) \leq 0.6$ yielded an optimal cochlea that

best fits physiological data [36].

The ear model described by Eqs. (2.2) to (2.7) is solved by applying initial and boundary conditions. The boundary conditions are

$$\left. \begin{aligned} \frac{\partial P}{\partial x} \Big|_{x=0} &= 2\rho C \frac{\partial^2 \xi_{ow}(t)}{\partial t^2} \\ P(L_{co}, t) &= 0 \end{aligned} \right\} \quad (2.8)$$

where $L_{co} = 3.5cm$ is the cochlear length, ξ_{ow} is the oval window displacement, and C_{ow} is the coupling factor of the oval window to the perilymph. In order to obtain ξ_{ow} , the middle ear model was applied [47] as expressed by the following differential equation:

$$\frac{d^2 \xi_{ow}(t)}{dt^2} + \gamma_{OW} \cdot \frac{d \xi_{ow}(t)}{dt} + \omega_{OW}^2 \cdot \xi_{ow}(t) = \frac{1}{\sigma_{OW}} \cdot [P(0, t) + \Gamma_{me} \cdot P_{in}(t)] \quad (2.9)$$

where σ_{OW}, γ_{OW} and ω_{OW} are oval window's areal density, resistance and resonance frequency. Furst 14, (At Section 2) shown The mechanical gain of the ossicles is denoted by Γ_{me} . the initial conditions are:

$$\left. \begin{aligned} \xi_{bm}(x, 0) &= \frac{\partial \xi_{bm}(x, t)}{\partial t} \Big|_{t=0} = 0; \\ \xi_{tm}(x, 0) &= \frac{\partial \xi_{tm}(x, t)}{\partial t} \Big|_{t=0} = 0; \\ \xi_{ow}(0) &= \frac{\partial \xi_{ow}(t)}{\partial t} \Big|_{t=0} = 0; \\ \psi_{OHC}(x, 0) &= \psi_0 \end{aligned} \right\} \quad (2.10)$$

2.2 Model of the Inner hair cell auditory nerve synapse

The basilar membrane motion is transformed into neural spikes of the auditory nerve by the inner hair cells. The deflection of the hair-cell stereocilia opens mechanically gated ion channels that allow any small, positively charged ions (primarily potassium

and calcium) to enter the cell [50]. Unlike many other electrically active cells, the hair cell itself does not fire an action potential. Instead, the influx of positive ions from the endolymph in the Scala media depolarizes the cell, resulting in a receptor potential. This receptor potential opens voltage-gated calcium channels; calcium ions then enter the cell and trigger the release of neurotransmitters at the basal end of the cell. The neurotransmitters diffuse across the narrow space between the hair cell and a nerve terminal, where they then bind to receptors and thus trigger action potentials in the nerve. In this way, the mechanical sound signal is converted into an electrical nerve signal. The IHC chronically leak Ca^{+2} . This leakage causes a tonic release of neurotransmitter to the synapses. It is thought that this tonic release is what allows the hair cells to respond so quickly to mechanical stimuli. The quickness of the hair cell response may also be due to that fact that it can increase the amount of neurotransmitter release in response to a change as little as $100\mu V$ in membrane potential.

Many models were developed for explaining the IHCs transduction abilities [46]. Some models focused on possible mechanisms for adaptation [57].

One commonly simplified modeling approach to explain the IHCs role in the auditory system posits a nonlinear system that combines AC and DC responses followed by a random generator that creates spike trains [57]. The model presented in this chapter is consistent with these principles.

Furst 14 calculates at Section 3 The BM displacement stimulates the IHC cilia to move, its velocity $\dot{\xi}_{ihc}$ corresponding to the BM velocity ($\dot{\xi}_{bm}$) by a nonlinear function,

e.g.

$$\begin{aligned}\dot{\xi}_{ihc} &= \alpha_1 \cdot \tanh(\alpha_2 \cdot \dot{\xi}_{bm}) \\ &\approx \alpha_1 \cdot \left[\alpha_2 \cdot \dot{\xi}_{bm} - \frac{(\alpha_2 \cdot \dot{\xi}_{bm})^3}{3} + \frac{2(\alpha_2 \cdot \dot{\xi}_{bm})^5}{15} \dots \right] \\ &\cong \dot{\xi}_{bm}\end{aligned}\tag{2.11}$$

Since the BM displacement in this model is nonlinear as described by the mechanical model above, we ignore the nonlinear terms in Eq. (2.11) and assume that $\alpha_1\alpha_2 = 1$ to approximate $\dot{\xi}_{bm}$;

The electromechanical receptors that are located in the IHC membrane yield an increase in the electrical potential (ψ_{ihc}) of the IHC membrane. A common modeling approach for the IHCs role in the auditory system is based on a nonlinear system that combines AC and DC responses. The DC level represents the firing responses without any synchrony to the input stimuli and the AC level represents the synchronized firing response (typical at low frequencies). The DC component includes a high-pass filter followed by a moving average filter of 2 ms long; the AC component consists of a low-pass filter. In order to account for physiological observations that demonstrated a reduction in synchronization as the frequency of the stimulus increases, a low-pass filter with a cutoff frequency of 300Hz was chosen with attenuation of 1800Hz. In practice, ψ_{ihc} is obtained by

$$\begin{aligned}\psi_{ihc}(x, t) &= e^{\gamma_{ihc}(x)} \cdot \left[\eta_{AC} \cdot \dot{\xi}_{ihc}(x, t) \star h_{ihc}(t) \right. \\ &\quad \left. + \eta_{DC} \cdot \int_{t-\delta}^t \{ \dot{\xi}_{ihc}(x, \tau) \cdot [1 - h_{ihc}(t)] \}^2 d\tau \right]\end{aligned}\tag{2.12}$$

where x represents the location of the IHC along the cochlear partition, $h_{ihc}(t)$ is the impulse response of the low-pass filter that represents the IHC response, and η_{AC} , η_{DC} , and Δ are constants Table A.1. The parameter $\gamma_{ihc}(x)$ represents the IHC efficiency index. It was defined as a function of x , to allow variability in IHC efficiency along the cochlear partition. For normal cochlea, we chose $\gamma_{ihc}(x) = 8$, which was found

to match experimental data. The efficiency of the IHC is reduced with a decrease of $\gamma_{ihc}(x)$.

This IHC receptor potential opens voltage-gated calcium channels; calcium ions then enter the cell and trigger the release of neurotransmitters at the basal end of the cell. The neurotransmitters diffuse across the narrow space between the hair cell and a nerve terminal where they then bind to receptors and thus trigger action potentials in the nerve.

The neural activity in the auditory system is irregular since a specific neuron might respond with a single spike or several spikes to a given stimuli. The origin of the stochastic activity of neurons is poorly understood. This activity results in both intrinsic noise sources that generate stochastic behavior on the level of the neuronal dynamics and extrinsic sources that arise from network effects and synaptic transmission. Another source of noise that is specific to neurons arises from the finite number of ion channels in a neuronal membrane patch. [14]

There are a number of different ways that have emerged to describe the stochastic properties of neural activity. One possible approach relates to the train of spikes as a stochastic point process. For example, studies suggested that the spontaneous activity of the cochlear nucleus can be described as a homogeneous Poisson process. Further investigations of the auditory system described the neural Cochlear Model for Hearing Loss response as a non-homogeneous Poisson point process (NHPP) whose instantaneous rate depends on the input stimuli. [14, 17, 37]

In the present chapter, we relate to the neural activity as NHPP, and thus only the

instantaneous rate (IR) should be extracted. In order to derive IR, we use the Weber-Fechner law, which describes the relationship between the magnitude of a physical stimulus and the intensity or strength that people feel. This kind of relationship can be described by a differential equation:

$$dP = K \frac{dS}{S} \quad (2.13)$$

2.2.1 ANR response

where dP is the differential change in perception, dS is the differential increase in the stimulus, and S is the stimulus at the instant. Integrating the above equation reveals $P = k \ln S + C$. Let us define $\lambda_{AN}(x, t)$ as the IR obtained by the auditory fiber attached to location x along the cochlear partition, and let us assume that it relates to the perception of the physical parameter. On the other hand, $\psi_{ihc}(x, t)$, the IHC electrical potential corresponds to the stimulus. Therefore, by applying the WeberFechner law, we obtained the relationship $\lambda_{AN}(x, t) = \ln(\psi_{ihc}(x, t)) + C$. However, the ANs IR should satisfy the following conditions: $0 < \lambda_{spont} \leq \lambda_{AN}(x, t) \leq \lambda_{sat}$, where λ_{spont} and λ_{sat} are the spontaneous and saturation rates of the AN, respectively. Therefore, $\lambda_{AN}(x, t)$ is obtained by

$$\lambda_{AN}(x, t) = \min\{\lambda_{sat}, \lambda_{spont} + \max\{0, A_{ihc}(x) * \ln(u(\psi_{ihc}(x, t)))\}\} \quad (2.14)$$

A_{ihc} is constant (see Table A.1)

In general, the auditory nerve response is divided into three types of fibers according to their spontaneous rates: a high spontaneous rate (HSR) that usually codes low-level stimuli, a medium spontaneous rate (MSR), and a high spontaneous rate (LSR)

that generally codes high level stimuli. In order to include all types of auditory nerves, we substitute in Eq. (13) the relevant constants $[\lambda_{spont}^{(H)}, A_H; \lambda_{spont}^{(M)}, A_M; \lambda_{spont}^{(L)}, A_L]$ for the HSR, MSR, and LSR that yield the instantaneous rates $[\lambda_{AN}^{(H)}(x, t), \lambda_{AN}^{(M)}(x, t), \lambda_{AN}^{(L)}(x, t)]$ respectively. The different types of ANs are distributed uniformly along the cochlear partition, with frequency of 61%, 23%, 16% to high medium and low rates.

2.3 Hearing Threshold, JND, Based on Auditory Nerve

The hearing threshold, defined as the lowest threshold of acoustic pressure sensation, is usually determined by quantitative psycho-acoustical experiments in which the human ability to detect the smallest difference in the stimulus physical property is obtained. This difference is referred to as a just-noticeable difference (JND). In such experiments, a subject must distinguish between two close time (t) dependent stimuli: $s(t, \alpha)$ and $s(t, \alpha + \Delta\alpha)$, where α is a given physical property. The $JND(\alpha)$ will be the minimum $\Delta\alpha$ a person can perceive. The parameter represents any physical property of the stimulus that can be measured such as frequency or level in monaural stimulus.

Comparing the behavioral JND and the neural activity is possible if one assumes that the neural system estimates the measured parameters. Siebert 41 obtained such a comparison when the JND of a single tones frequency and level was compared to the neural activity of the auditory nerve. Sieberts findings were based on the assumption that the auditory nerve (AN) response behaves as an NHPP, and the brain acts as an unbiased optimal estimator of the physical parameters. Thus, the JND is equal to the standard deviation of the estimated parameter and can be derived by lower bounds such as the CramerRao lower bound. Heinz and Carney 19 generalized Siberts results

to a larger range of frequencies and levels.

In a psychoacoustical JND experiment, the yielded JND value is obtained when $d' = 1$, which is expressed by Furst 14, Section 4:

$$d' = \frac{E[\hat{\alpha} | \alpha^*] - E[\hat{\alpha} | (\alpha^* + \Delta\alpha)]}{std(\hat{\alpha} | \alpha^*)} = \frac{\Delta\alpha}{std(\hat{\alpha} | \alpha^*)} \quad (2.15)$$

where $E[\hat{\alpha} | \alpha^*] = \alpha^*$, α^* is the true value of α , and $\hat{\alpha}$ is the estimated value of α .

Therefore, $d' = 1$, yields the relations $\Delta\alpha = std(\hat{\alpha} | \alpha^*)$, which implies

$$JND(\alpha^*) = std(\hat{\alpha} | \alpha^*) \quad (2.16)$$

When the estimation is based on neural activity that behaves as NHPP, there are two possible ways to analyze the performance. The first way is referred to as "rate coding" (RA), which means that the performance is analyzed on the basis of the number of spikes. The second way is referred as "all information coding" (AI), indicating that in addition to the number of spikes in the interval, the timing of the discharge spikes is considered as well.

Let us define $N(0, T)$ as the random variable that represents the number of spikes in the time interval $[0, T]$. For the RA coding, the probability density function (pdf) of getting n spikes in the time interval of length T is obtained by

$$P_{RA}(N(0, T) = n) = \frac{1}{n!} \cdot \left[\int_0^T \lambda(t, \alpha) dt \right]^n \cdot \exp \left\{ - \int_0^T \lambda(t, \alpha) dt \right\} \quad (2.17)$$

where $\lambda(t, \alpha)$ is the instantaneous rate of the nerve fiber that depends on both the time t and the physical parameter α . Given the RA pdf (Equation (2.17)), the resulting

CramerRao lower bound (CRLB) is obtained by [3]

$$CRLB_{RA}(\alpha^*) = \left\{ \frac{T}{\bar{\lambda}(\alpha^*)} \left[\frac{\partial \bar{\lambda}(\alpha^*)}{\partial \alpha} \Big|_{\alpha=\alpha^*} \right]^2 \right\}^{-\frac{1}{2}} \quad (2.18)$$

$\bar{\lambda}(\alpha^*) = \frac{1}{T} \cdot \int_0^T \lambda(t, \alpha) dt$ is average rate for interval $[0, T]$.

For the AI coding, the probability density function of getting n successive neural spikes at a set of time instances is t_1, t_2, \dots, t_n , where $0 \leq t_1 < t_2 < \dots < t_n \leq T$ is obtained by

$$P_{AI}(N(0, T) = n, t_1, \dots, t_n) = \frac{1}{n!} \cdot \prod_{k=1}^n \lambda(t_k, \alpha) \cdot \exp \left\{ - \int_0^T \lambda(t, \alpha) dt \right\} \quad (2.19)$$

and the CRLB yields

$$CRLB_{AI}(\alpha^*) = \left\{ \int_0^T \frac{1}{\lambda(t, \alpha^*)} \left[\frac{\partial \bar{\lambda}(\alpha^*)}{\partial \alpha} \Big|_{\alpha=\alpha^*} \right]^2 \right\}^{-\frac{1}{2}} \quad (2.20)$$

For the unbiased system, the rule is

$$std(\hat{\alpha} | \alpha^*) \geq CRLB_{RA}(\alpha^*) \geq CRLB_{AI}(\alpha^*) \quad (2.21)$$

In an optimal unbiased system, the standard deviation of the estimator can achieve the lower bounds. Since $JND(\alpha^*) = std(\hat{\alpha} | \alpha^*)$ (Equation (2.16)), $JND(\alpha^*)$ can be estimated by calculating $CRLB_{RA}(\alpha^*)$ or $CRLB_{AI}(\alpha^*)$. Comparing the estimated thresholds to experimental results can resolve the question whether the brain estimates the auditory thresholds according to RA or AI coding.

In order to apply the above-mentioned method for determining the auditory threshold, we should consider the responses of all 30,000 AN fibers that innervate each ear.

Since the AN fibers are statistically independent [20], the d theorem can be applied, which yields

$$(d')^2 = \sum_{m=1}^M (d'_m)^2 \quad (2.22)$$

where M is the number of nerve 1 fibers and d'_m is the d (Equation (2.15)) that was derived for the m th fiber. Moreover,

$$std(\hat{\alpha} | \alpha^*) = \frac{1}{\sum_{m=1}^M [std_m(\hat{\alpha} | \alpha^*)]^{-2}} \quad (2.23)$$

where $std_m(\hat{\alpha} | \alpha^*)$ is the standard deviation of the estimator obtained by the m th fiber. Since the threshold is obtained when $d' = 1$, it implies that in an optimal system,

$$JND(\alpha^*) = \frac{1}{\sqrt{\sum_{m=1}^M [CRLB_m(\alpha^*)]^{-2}}} \quad (2.24)$$

where $CRLB_m(\alpha^*)$ is the CRLB of the m th fiber.

Let us define the number of fibers attached to each location along the cochlear partition as $M(x)$. Thus, $\int_{x \in [0, L_{co}]} M(x) dx = 30,000$, where L_{co} is the cochlear length. For every location, three IRs were derived $\lambda_{AN}^{(H)}(x, t), \lambda_{AN}^{(M)}(x, t), \lambda_{AN}^{(L)}(x, t)$ Eq. (2.14), which correspond to the HSR, MSR, and LSR fibers, respectively. They are distributed uniformly along the cochlear partition with corresponding weights w_L, w_M, w_H (see Table A.1). Therefore,

$$JND(\alpha^*) = \frac{1}{\sqrt{F_H + F_M + F_L}} \quad (2.25)$$

Such that

$$\left. \begin{aligned} F_H &= \sum_{x \in [0, L_{co}]} \sum_{m=1}^{\omega_H \cdot M(x)} \left\{ CRLB_m^{(H)}(\alpha^*) \right\}^{-2} \\ F_M &= \sum_{x \in [0, L_{co}]} \sum_{m=1}^{\omega_M \cdot M(x)} \left\{ CRLB_m^{(M)}(\alpha^*) \right\}^{-2} \\ F_L &= \sum_{x \in [0, L_{co}]} \sum_{m=1}^{\omega_L \cdot M(x)} \left\{ CRLB_m^{(L)}(\alpha^*) \right\}^{-2} \end{aligned} \right\} \quad (2.26)$$

Replacing CRLB in Eq. (2.25) with the corresponding $CRLB_{RA}(\alpha^*)$ or $CRLB_{AI}(\alpha^*)$, $JND(\alpha^*)$ is estimated by either RATE or AI coding.

In order to calculate both $CRLB_{RA}(\alpha^*)$ or $CRLB_{AI}(\alpha^*)$, the derivative of the instantaneous rate should be derived. We have used the following approximation:

$$\partial \lambda(t, \alpha) / \partial \alpha|_{\alpha=\alpha^*} \approx \frac{\lambda(t, \alpha^* + \Delta \alpha) - \lambda(t, \alpha^*)}{\Delta \alpha} \quad (2.27)$$

Therefore, in deriving $JND(\alpha^*)$ for any stimulus $s(t, \alpha^*)$, the IRs for both stimuli $s(t, \alpha^*)$ and $s(t, \alpha^* + \Delta \alpha)$ should be calculated. Two types of thresholds will be presented for tones in quiet and in the presence of noise. The quiet threshold was derived by substituting $\alpha^* = 0$ that yielded $\lambda(t, \alpha^*) = \lambda_{spont}$. For the thresholds in the presence of noise, $s(t, \alpha^*)$ is equal to the noise, and $s(t, \alpha^* + \Delta \alpha)$ is equal to the noise +tone with a level of $\Delta \alpha$.

Chapter 3

GPU With CUDA Architecture Updates

GPU is specialized computer for fast image processing and construction for display. However more than decade ago NVidia start to adjust the architecture for general applications that can be parallelize. the GPU consists of multiple clusters of core arrays, each core array has shared cache memory and registers for execution. this architecture allow to run program in SIMD (Same Instruction Multiple Data), each clock cycle all active cores run the same operation for different memory parts. NVidia architecture divided to 3 types

1. Global is accessible for all Streaming multiprocessor (clusters of core arrays), this make access relatively slow, hundreds of cycles per read or write, however execution control can switch between active cores while memory is loaded and conceal the delay.
2. Cache is accessible for each Multiprocessor separately, is order of magnitude faster to access but is much smaller than the Global Memory, its advised to use this for synchronizing data between threads of the same block or overflow of data that cant be stored in the registers.

3. Register File are very limited amount of memory close for the processors, they are accessible in single clock cycle, all instructions executions can be done there.
4. Constant Memory, small amount of it accessible for each core array separately, while this is read only memory when executing code from the device, the host can upload data before kernel execution. this memory can be accessed fast if all threads in warp access the same cell at clock cycle, otherwise, it will be serialized and slow the read operation by number of different cells needed to be accessed.

CUDA (Compute Unified Device Architecture) has 2 contexts of execution, host is the CPU, all CUDA programs starts with host execution, control all memory copy to the GPU and from the GPU and launches the device (GPU) context procedures called kernels. each kernel has 2 level hierarchy

1. Blocks each block run on single multiprocessor and has access to the same cache memory, each multiprocessor can run multiple Blocks but they order of execution is not guaranteed. while multiple blocks can communicate through the atomic actions, its slow and mostly unadvised, since core array need to stop and switch execution to another block or stop completely if all blocks are waiting
2. Threads, each Block can contain up to 1024 they execute concurrently (32 of them each cycle), they can access the same share memory array with few limitations. each time threads need to read memory that need multiple clock cycle to be accessible the multiprocessor can switch context and make this threads dormant, this is fast operation.

all Threads on each block has access to the same shared memory array (SHM), which defined on cache by the programmer. SHM access is divided to banks of either 4 bytes or on some cards 8 bytes. each thread can only access single bank, if n threads read or write to distinct n banks, all can be serviced simultaneously which multiply memory bandwidth by n . but if those threads read/write to the same bank, a conflict is occurred, the chip will split the memory access to n free conflict serialized requests. SHM is placed in L1 cache memory on chip is much faster to deliver requests (1 clock cycle) than either the local or global memory, but n way bank conflict will reduce access speed by n factor and should be avoided. if multiple threads of the same warp read single bank, multi-casting occur and all threads gets memory at the same time, for multi-write to the same bank, only a single thread will write, arbitrary selected.

3.1 Difference Between Generations of NVidia architecture

SM basic properties While the Streaming Multiprocessor (SM) is the center of the GPU and controls execution of the kernel. NVidia has changed over the years components included in SM. each compute capability is identified with SM model. this project optimized and run on several capabilities.

Compute Capability 1.x, Tesla Architecture NVidia architectures is divided to generations of processors (identified by the number before decimal point, major version). the first generation architecture 1.x [31], presented maximum capacity of 768 threads per SM, had only 8 FP32 lane and 2 SFU for single and 1 double precision unit. since threads executed in groups of 32 called warps, each multiply/add command took 4

clock cycles to complete and intrinsic functions such as *fdividef*, *logf* will take 16 cycles to complete. our project needs 256 parallel threads per block to represent spatial partition [40], this limits us to maximum 3 blocks on 1.0-1.1 and 4 on 1.2-1.3. another critical resource is 32 bit registers per SM, with each has only 8K-16K, for maximum capacity each thread would have to use 10-16 registers, effectively we can expect at most single block on 1.0-1.1 or 2 blocks on 1.2-1.3 (32 registers per thread), this significantly limit the parallelism of 1.x SMs relative to later generations. shared memory is also limiting factor, architecture 1.x had 16 banks per warp. reading shared memory took 2 cycles per warp, bank conflicts must be avoided between 2 halves of the warp. this model has been deprecated and is not supported by CUDA.

Compute Capability 2.x, Fermi Architecture NVidia second generation of architecture significantly modified relative to the first. Comparison Tables B.1 and B.2, for SM 2.0 doubling the amount of registers per SM to 32K and increase warps per SM to 48 allows more active threads, there are only 4 SFU for every 32 FP32 lane meaning that intrinsic will take 8 cycles, single precision multiply, add and multiply-add take 1 clock cycle to execute. With the number of FP32 lanes multiplied by 4 and registers 2, register pressure increased, this phenomena consists reducing occupancy of kernel due to not enough registers for the maximum number of threads, example, SM 2.0 can run 1536 simultaneously, if each requires 32 registers maximum occupancy will achieved with use of 48K, but SM 2.0 has only 32K so maximum occupancy cant rise above 66%. The 2nd generation architecture also doubled the number of warp schedulers, while occupancy defined amount of warps hold in SM registers at once, each sched-

uler execute warp if enough FP32 lanes available, which happens when warp need to read from memory and is stalled. SM 2.1 increased number of FP32 lanes to 48 and SFU units to 8, allowing also for 2 instructions to execute per warp, if they are independent. this configuration while increase register pressure improves throughput for arithmetic intense programs as well as intrinsic operations. this model also deprecated since CUDA 8 and is not supported.

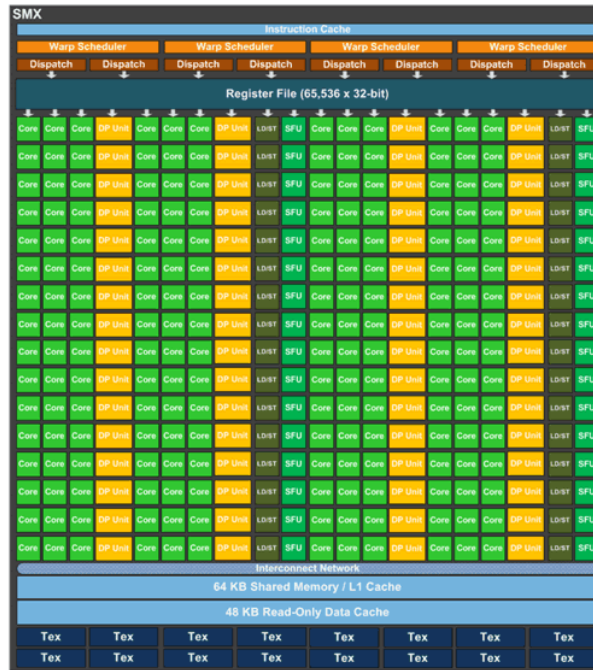
Compute Capability 3.x, Kepler Architecture Third generation of CUDA supported architecture (SM 3.x, called SMX), shown in Fig. 3.1 left side. Increased amount of FP32 lanes to 192 (light green boxes), and SFU to 32 (forest green boxes), doubling number of warp schedulers to 4 (orange boxes on top), which improves the ratio of FP32 lanes per scheduler allowing more arithmetic intense programs. increasing maximum threads per SM to 2048 while doubling registers number to 64K (Dark blue box under brown boxes) Reduce register pressure permit 32 per thread even for maximum occupancy. the number of double precision units per SM is only 6 (yellow boxes, note since the figure shows the Quadro variant of the core described at Section 3.1 we see 64 of them, on Geforce there are only 8 per SMX), this makes 64-bit operations to be avoided, however the 2 instructions per scheduler (brown boxes are dispatchers) expanded to execute 64 bit instruction with 32-bit one if independent, Kepler's schedulers assigned . Each SMX can read/write 128 byte (Olive boxes can read 4 Bytes each) architecture 3.5 added dynamic parallelism [33], this feature allow kernel launching from kernels, this improve speeds by avoiding copy kernel data and instructions from host to device for run time decisions. the launched kernel called child, and is guaranteed to execute

and return before parent completion. the Kepler Cache memory divided to L1, allocated per SMX and L2 cache shared for all SMX. the L1 has 64KB (Cyan box near the bottom) and can be configured as 48/16, 32/32 or 16/48 KB between L1 (used for register spilling) and shared memory, allocated per block must be reduced, or CUDA will reduce occupancy so each remain block will have enough shared memory.

Compute Capability 5.x and 6.1-6.2, Maxwell and Pascal Architecture The Maxwell SM (CC 5.x), designated SMM, has been restructured to improve energy efficiency, [32], shown in Fig. 3.1 right side. The colored boxes functions identical to those in Kepler's SMX chart. The SMM divided to 4 warp schedulers, each can dispatch 1 instruction per cycle (Table B.2), each physically connected to its 32 FP32 lanes, 8 SFU and single DP unit. this doubles energy efficiency and allows NVidia to also double numbers of SM with decrease in energy consumption and die size increase from $294mm^2$ to $398mm^2$. Note that while number of instructions per cycle executed reduced from 2 to 1, number of ALUs per SM decreased too, so more SM per Card can be set (on comparable tiers of cards). With both architectures limited to 2048 threads per SM and 64K registers per SM, Maxwell architecture can run double number of threads on single card. note however that count of DP units/FP32 lanes lowered from 1/24 in Kepler to 1/32 (this relevant for Geforce, Section Quadro and Tesla series Differences from GeForce), make the use of double precision inadvisable. Maxwell Architecture also introduce separation between L1 cache (24KB) and shared memory (64KB in 5.0, 96K from 5.2, Table B.1). In difference from Kepler unified 64KB that can be configured between the shared and L1 cache. Pascal, CC 6.1-6.2, designated SMP, has mostly same

features as the CC 5.2 with increase in L1 Cache from 24 to 48KB, this allows to better ratio of cache hits/miss from global memory. register spills are stored on L2 from Maxwell, not L1 like in Kepler and this need to be considered when limiting number of registers per thread.

NVIDIA Kepler SM



NVIDIA Maxwell SM

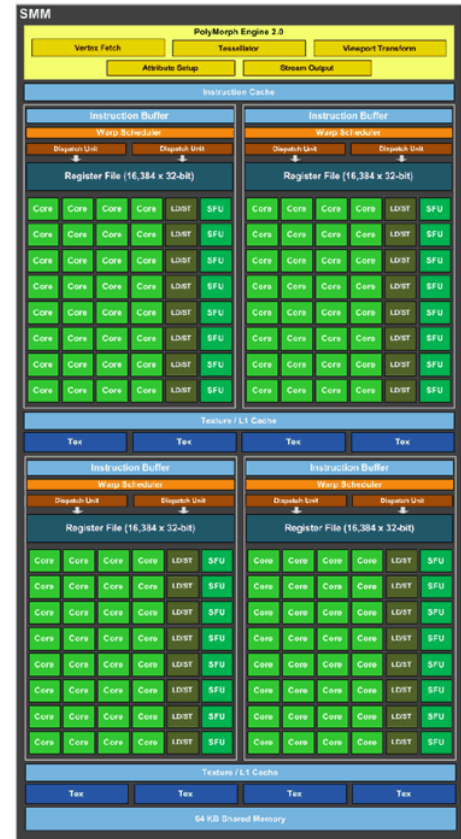


Figure 3.1: Comparison of Kepler (CC 3.x) and Maxwell (5.x) SM based on figures from [30] and [32]

Quadro and Tesla series Differences from GeForce NVidia manufacture several variations for each CC version, this versions of SMs are differed by the number of DP units on SM. numbers in previous paragraphs described GeForce model, which is available for this project, for Quadro and Tesla Versions there is 1/2 ratio of DP units/FP32 lanes in Fermi,Maxwell and 1/3 ration in Kepler (available in CC 3.7). the ratio

is 1/2 on Pascal 6.0 as well, however CC 6.0 has SM with 64 FP32 lanes and 2 warp scheduler, with double number of SM per FP32 lane and limitation of 2048 threads per SM, CC 6.0 can run double number of threads per lane if registers count is below 16 per thread.

3.2 NVidia Nsight

The NVidia Nsight tool allow for both CUDA debugging and performance analysis [2]. Since our project aim is to accelerate solution of the Cochlear equations, Nsight enable us to test multiple aspects of card workload and find bottlenecks. We present here the tests used to optimize performance in this project.

Achieved Occupancy describes actual number of warps that are executed for the current kernel on the SM, Kepler Architecture can run 64 warps on each SM in granularity of blocks, if kernel needs between 49 to 64 registers per thread and block contains 256 threads, no more than 1024 will be able to run (4 blocks) per SM and Achieved Occupancy will be $\frac{1024}{2048} = 50\%$. its advisable to feed amount of block that will be larger than achieved occupancy times number of SM and will divide it as well, to prevent small number of last blocks being processed while most of the SM idle, called Tail End effect.

Achieved FLOPs describes the number of floating point operations per second executed on the device. its common method evaluate performance which can be assembled from several independent sequential tasks that run on multi-core or any device with shared memory and can be adjust for use of cores with different operation fre-

quency [23], this metric is useful for optimization process of time-to-solution.

Achieved IOPs indicates number of Integer Operations per second executed on the card, while similar to Achieved FLOPs this methodology is useful for differentiating between solutions performance and not to derive the causes of bottlenecks.

Instruction Statistics is basic measurement of the global work distribution scheduler and has several indicators, Instructions Per Cycle (IPC) is measured average for warps across the SM. Issued IPC describe the amount of dispatched instructions for the warp and will be ideally identical to the Executed IPC, however some instructions has to be issued multiple times. Instructions Per Warp (IPW) shows average executed per warp/cycle. Warps Launched, per SM. if this number varies greatly, most likely cause is insufficient warps to get Achieved Occupancy across all SM, called tail effect and reduce average time-to-solution across the device due to idle warp scheduler that wait for remained active warps to finish kernel.

Issue Efficiency measures the ability of the device to allocate instruction for execution per cycle, composed from several indicators, Active Warps measured for each SM and averages number of allocated warps for the SM across kernel, this can't be higher than Achieved Occupancy but if significantly lower there aren't enough warps to exploit device parallelism, if varied greatly some of SMs finish before others and waiting idle, should consider workload redistribution. the eligible warps is average number of warps that can execute instruction each cycles, this number can't be higher than warp schedulers count per SM. warp issue efficiency is the percentage of time that scheduler can issue instruction for one of its warps, if too

low, the graph of issue stall reasons indicates division of failures for schedulers to allocate instructions, such as memory dependency, waiting for load/store units to be available to read/write to memory. execution dependency, data for executed instruction is not available etc.

Memory Statistics measures number of requests and bytes across all memory modules, includes shared, constant, local and global memory as well as L1, L2 cache. this can indicate bottleneck of stalls issues due memory calls.

Source Level Experiments offer several indicator in resolution of single instruction, include total number of times executed in the kernel, percentage of threads inside warp that executed the instruction.

Occupancy Calculator This NVidia tool is an Excel sheet that can feed the CC, threads per block for kernel, registers per thread and SHM to calculate maximum blocks per SM. this tool gives target registers number to increase occupancy.

Chapter 4

Cochlear Model Implementation Updates

While Sabo et al. 39 Shown that the Cochlear Model Equations can be approximated using NVidia GPU. An updated version of the Hardware require some optimizations to fit for updated version of CUDA supporting devices

4.1 Single precision computations updates

4.1.1 Serial Solution of boundary conditions equations

Sabo et al. 39 get boundary conditions from partial difference equation Eq. (2.1) by substituting Eqs. (2.2) and (2.3) yields:

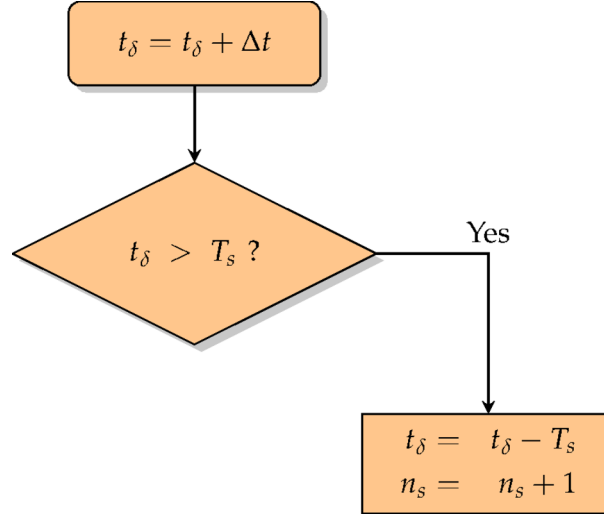
$$\frac{\partial P(x, t)}{\partial x^2} = Q(x) \cdot [P(x, t) - G(x, t)] \quad (4.1)$$

$$G(x, t) = - [R_{bm}(x)\xi_{bm}(x, t) + K_{bm}(x)\dot{\xi}_{bm}(x, t) + P_{tm}(x, t)] \quad (4.2)$$

$$Q(x) = \frac{2 \cdot \rho \cdot \beta}{A \cdot M_{bm}(x)} \quad (4.3)$$

With t as parameter Eq. (4.1) is second order regular differential equation and is depend on x it can be solved by applying boundary conditions from Eqs. (2.8) and (2.9)

The boundary condition problem is solved for every time point by converting the problem from the continuous to the discrete domain. The cochlear length is divided

Figure 4.1: Flow chart to replace $t + \Delta t$.

into N equal sections where $x = n \cdot \Delta x$ for $0 \leq n \leq N$ and

$\Delta x = \frac{L_{co}}{N}$. The first and second derivatives of P are approximated using the Taylor

series in the discrete domain:

$$\frac{\partial P(x_n, t)}{\partial x} \approx \frac{P(x_{n+1}, t) - P(x_n, t)}{\Delta x} \quad (4.4)$$

$$\frac{\partial^2 P(x_n, t)}{\partial x^2} \approx \frac{P(x_{n+1}, t) - 2 \cdot P(x_n, t) + P(x_{n-1}, t)}{\Delta x^2} \quad (4.5)$$

substitute Eq. (4.4) with first boundary condition in Eq. (2.8) for $x = 0$ yields

$$P(\Delta x, t) = P(0, t) + \Delta x \cdot 2\rho \cdot C_{ow} \cdot \frac{\partial^2 \xi_{ow}(t)}{\partial t^2} \quad (4.6)$$

$\frac{\partial^2 \xi_{ow}(t)}{\partial t^2}$ is obtained from Eq. (2.9) modifies Eq. (4.6) to

$$P(\Delta x, t) = (1 + \Delta x \cdot \frac{2\rho \cdot C_{ow}}{\sigma_{ow}}) \cdot P(0, t) + Y_0 \quad (4.7)$$

where

$$Y_0 = 2\rho C_{ow} [\frac{\Gamma_{ME}}{\sigma_{ow}} P_{in}(t) - \gamma_{ow} \dot{\xi}_{ow}(t) - \omega_{ow}^2 \xi_{ow}(t)] \Delta x \quad (4.8)$$

The second boundary condition of Eq. (2.8) for $x = L_{co}$ yields

$$P(x_N, t) = 0 \quad (4.9)$$

substituting Eq. (4.5) in Eq. (4.1) for $1 \leq n \leq N - 1$ yields

$$P(x_{n+1}, t) - (2 + \Delta x^2 \cdot Q(x_n)) \cdot P(x_n, t) + P(x_{n-1}, t) = \Delta x^2 \cdot Q(x_n) \cdot G(x_n, t) \quad (4.10)$$

Equations (4.7), (4.9) and (4.10) can be expressed as matrix form

$$\Lambda \underline{P}(t) = \underline{Y}(t) \quad (4.11)$$

where Λ is tridiagonal matrix with main diagonal

$$\left[-(1 + \Delta x \cdot \frac{2\rho \cdot C_{ow}}{\sigma_{ow}}), -(2 + \Delta x^2 \cdot Q(x_1)), \dots, -(2 + \Delta x^2 \cdot Q(x_{N-1})), 1 \right] \quad (4.12)$$

and the two other non-zero diagonals are ones, except for the (N,N1) element of the matrix, which is zero. $\underline{P}(t)$ and $\underline{Y}(t)$ are shown below.

$$\underline{P}(t) = [P(0, t), P(x_1, t), \dots, P(x_{N-1}, t), P(x_N, t)]^T \quad (4.13)$$

$$\underline{Y}(t) = [Y_0, \Delta x^2 Q(x_1) G(x_1, t), \dots, \Delta x^2 Q(x_{N-1}) G(x_{N-1}, t), 0]^T \quad (4.14)$$

given $\underline{Y}(t)$ the solution of Eq. (4.11) is pressure of cochlear partition

$$\underline{P}(t) = \Lambda^{-1} \underline{Y}(t) \quad (4.15)$$

The inversion of the matrix Λ was obtained by LU decomposition.

4.1.2 Serial Solution of initial conditions equations

A set of initial conditions of second order ordinary differential equations for $\xi_{bm}(x, t)$, $\xi_{tm}(x, t)$, $\xi_{ow}(t)$ and a first order differential equation of $\psi(x, t)$ as a function of t is obtained by Eqs. (2.5) and (2.9) along with the substitution of Eqs. (2.2) and (2.7), and Eq. (2.6) in Eq. (2.3). This set is solved by using the modified Euler method for first order initial condition equations with adaptive step size [28]. The second order differential equations are solved by defining the first derivative of $\xi_{bm}(x, t)$, $\xi_{tm}(x, t)$, $\xi_{ow}(t)$ as additional unknown variables $\dot{\xi}_{bm}(x, t)$, $\dot{\xi}_{tm}(x, t)$, $\dot{\xi}_{ow}(t)$. Thus five first order differential equations have to be solved for every point (x) along the cochlear partition and two additional first order differential equations for $\xi_{ow}(t)$ and $\dot{\xi}_{ow}(t)$. If N is the number of samples along the cochlear partition, there are a total $5N + 2$ initial conditions to solve. The solution algorithm is explained shortly as follows. Let's define a typical initial condition problem as:

$$\left. \begin{aligned} \frac{dy}{dt} &= f(t, y(t)) \\ y(0) &= Y_0 \end{aligned} \right\} \quad (4.16)$$

the solution use Euler method and yields for step size Δt

$$y(t + \Delta t) = y(t) + \Delta t \cdot f(t, y(t)) \quad (4.17)$$

In particular, for $t = 0$, $y(\Delta t) = y_0 + \Delta t \cdot f(0, y_0)$:

However, when the Euler method is used convergence is ensured only when

Δt is very small (in the order of 10^{-15} s). Such a step size is time consuming and unpractical. In order to increase the step size the modified Euler method was used. An iterative series $\{\omega_n\}$ is defined for deriving $y(t + \Delta t)$ for a known $y(t)$:

$$\left. \begin{aligned} \omega_0 &= y(t) + \Delta t \cdot f(t, y(t)) \\ \omega_n &= y(t) + \frac{\Delta t}{2} \cdot [f(t, y(t)) + f(t + \Delta t, \omega_{n-1})], n \geq 1 \end{aligned} \right\} \quad (4.18)$$

The first element in the series is obtained by the Euler method. If the iterative series is convergent, then $\omega_n \rightarrow y(t + \Delta t)$. The condition for convergence can be determined by the Lipschitz constant for a function $f(t, y(t))$ that obeys the condition:

$$|f(t + \Delta t, y(t + \Delta t)) - f(t, y(t))| \leq L|y(t + \Delta t) - y(t)| \quad (4.19)$$

where L is Lipschitz constant. By substituting Eqs. (4.17) and (4.19) and replacing ω_{n-1} and ω_n by $y(t + \Delta t)$ in Eq. (4.18) Sabo et al. 39 get constraint

$$\frac{L \cdot \Delta t}{2} < 1 \quad (4.20)$$

thus, when $\Delta t < 2/L$. ω_n will converge to $y(t + \Delta t)$. Since it is rather complicated to evaluate the Lipschitz constant for the set of the differential equations described above, an estimate for the Lipschitz constant was derived from Eq. (4.4) as follows [28]

$$\hat{L} = \frac{|f(t + \Delta t, \omega_n) - f(t, \omega_{n-1})|}{|\omega_n - \omega_{n-1}|}, n \geq 1 \quad (4.21)$$

In order to solve $5N + 2$ initial conditions, time step Δt must satisfy Lipschitz constraint Eq. (4.20) for every equation. To lower computation load, Sabo et al. 39 chose to verify the most sensitive variable, ξ_{bm} , such that $\Delta t < 2/L$ for N equations. it means

that the existing time step, Δt , was sufficient for convergence thus the resulted derivation was stored. The size of the time step was multiplied by 2 if $\max_N \{ \frac{\hat{L}\Delta t}{2} \} < 0.25$. when $\hat{L}\Delta t \geq 1$, time step Δt was too large for convergence and was divided by 2 and procedure repeated with the smaller time step.

4.1.3 Run Time Estimation

Sabo et al. [39] algorithm started by set time t and variables $\xi_{bm}, \xi_{tm}, \xi_{ow}, \psi$ to 0 to fulfill initial conditions at Eq. (2.10), then compute $\underline{Y}(t)$. The next phase is deriving $\underline{P}(t)$ by solving equations system (4.15) by the LU decomposition and then update t to $t + \Delta t$ and start next iteration. The input of the algorithm is acoustic stimulus of $P_{in}(t)$. F_s denotes sample frequency of the system, thus the interval time between consecutive samples is $T_s = \frac{1}{F_s}$. Since the algorithm convergence requirements yields higher time resolution then the input signal, linear interpolation is done in order to compute the input pressure for time t as follows:

$$\begin{aligned} n_s &= \left\lfloor \frac{t}{T_s} \right\rfloor, \Delta\tau = \frac{t - n_s \cdot T_s}{T_s}, \\ P_{in}(t) &= (1 - \Delta\tau) \cdot P_{in}(n_s \cdot T_s) + \Delta\tau \cdot P_{in}((n_s + 1) \cdot T_s) \end{aligned} \quad (4.22)$$

For convergence of the partial differential equation and according to CFL condition Courant et al. [7] and given $N = 256$, condition $\frac{c\Delta t}{\Delta x} < 1$ when c is speed of sound inside perilymph. Given a cochlea length of $0.035m, c = 1500m/s$ condition met when $\Delta t < 9.11 \cdot 10^{-8}$. In the simulations, the typical time step was in the range of $1 \cdot 10^{-7}$ to $1 \cdot 10^{-7}$ and maximum value was $1 \cdot 10^{-6}$ with 6 iteration guarantee convergence. [39] calculates 226 instructions per iteration with $2.5 \cdot 10^{-7}s$ as average calculated for $N = 256$ cochlear sections, estimated execution time for 1 second of acoustic simulation is $\frac{10^7}{2.5} \cdot 6 \cdot 226 \cdot 256 = 1.388 \cdot 10^{12}$ clock cycles, given 3 – 4GHz CPU, about 350 to 500

seconds execution time for 1 second of stimulus.

4.2 Parallelizing the algorithm

Serial solution of the algorithm depends on LU decomposition in the longitudinal dimension and iterative steps on time dimension. This methods takes duration which is several orders of magnitude longer than real time voice signal. Sabo et al. 39 developed massive parallel algorithm solution which efficiently use massive parallelism of GPU.

4.2.1 Parallelism in time dimension

The model uses a one dimensional description of the cochlear partition, hence it was natural to chose a one dimensional grid in the parallel implementation. The cochlear partition was divided into N_x sections along the x-axis, each section processed by one thread. The parallel algorithm uses large amounts of constant data in its computations. The data are read from the CPU host to the global memory of the GPU during the initialization phase of the launch of the CUDA kernel. At the end of a time step, when that time step converges, the algorithm stores the current results and advances to the next time step. The memory traffic can slow down the execution time dramatically. To prevent excessive memory accesses the equation is solved for a long time interval in one kernel launch, so that the traffic to and from the global memory is done once for a long segment of input processing. Each thread handles one cochlear section and resolves the basilar membrane velocity for block time interval t_{bti} seconds. The thread stores results to global memory once in each Δt_{OUT} seconds, where Δt_{OUT} represents

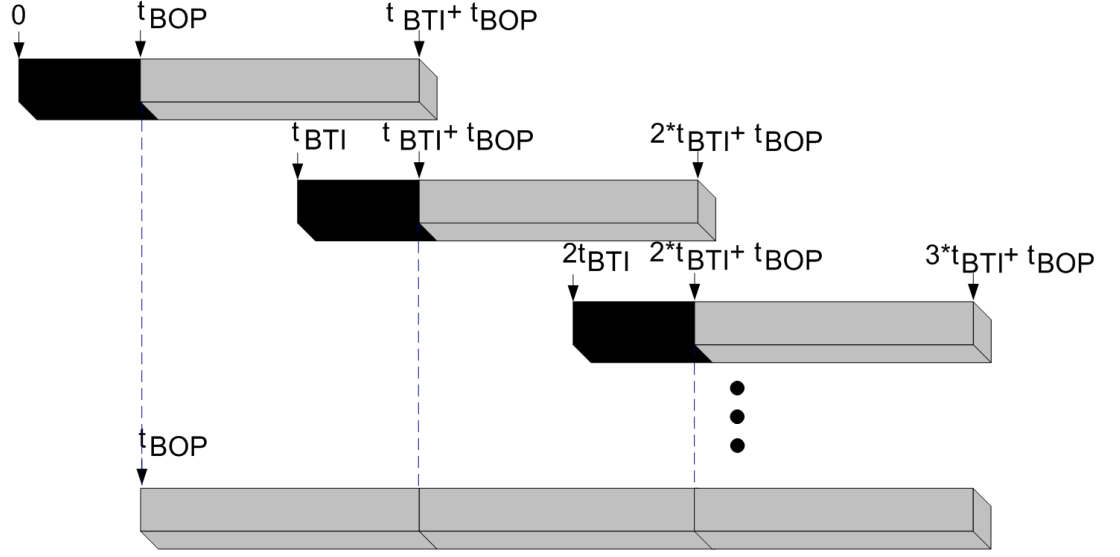


Figure 4.2: Partitioning in the time dimension each bar represents a time interval handled by one CUDA block. Overlapping sections are discarded

the desired temporal resolution of the output.

To achieve parallelization in the time dimension Sabo et al. 39 partitioned the processing into N_t CUDA blocks, such that each block solves the equations for all the cochlear sections for a different time interval. Hence the number of time intervals is equal to the number of CUDA blocks. The time intervals are handled by blocks that are not completely independent, and two consecutive time intervals, mapped to two consecutive CUDA blocks, have an overlapping interval of t_{BOP} seconds. This requires each block to run the algorithm for a total of $t_{BTI} + t_{BOP}$ seconds. That is, a specific block BL solves the model for the interval $T^{(BL)} = [(BL - 1) \cdot t_{BTI} \leq t \leq BL \cdot (t_{BTI} + t_{BOP})]$, Since the output is required to have a temporal resolution of Δt_{OUT} , for each kernel launch each thread returns $\frac{t_{BTI} + t_{BOP}}{\Delta t_{OUT}} + 1$ results of a specific cochlear section. The results computed in the first t_{BOP} seconds for each block are discarded. Figure 4.2 illustrates the time dimension partitioning graphically, showing the time

interval processed by each block, including the discarded (overlapping) interval. The non-overlapping intervals processed by each block are connected together to form the final result.

4.2.2 Parallelism in longitudinal dimension

Sabo et al. 39 found that the boundary conditions described at Eq. (4.10) are the main obstacle for parallelism due to inter dependency between the sections. the serial solution use LU decomposition to solve the equation. However this is not good solution for massive parallel algorithm, hence incompatible for GPU.

Basic smoothers (numerical algorithms used for solving a system of linear equations) like the Jacobi method and SOR (successive over relaxation) can be used to parallelize a linear system. The purpose of smoothers is to reduce, or smooth, in an efficient way the approximation error [53]. More advanced smoothers are difficult to solve in parallel [16].

Taking advantage of the tridiagonal shape of the equations set, we used Jacobi relaxation and implemented it with only a few floating point operations for each Jacobi iteration. Manipulating Eq. (4.10) yields for $1 \leq n \leq N - 1$

$$\left. \begin{aligned} \hat{Q}(x_n) &= \frac{1}{-(2+\Delta x^2 Q(x_n))} \\ P(x_n, t) &= (Y_n(t) - 1 \cdot P(x_{n-1}, t) - \cdot P(x_{n+1}, t)) \cdot \hat{Q}(x_n) \end{aligned} \right\} \quad (4.23)$$

by denoting the following

$$\begin{aligned} \hat{A}_l &= [0, 1, \dots, 1, 0]^T \\ \hat{A}_u &= [1, 1, \dots, 1, 0]^T \end{aligned}$$

and

$$\hat{A}_m = [\frac{1}{-(1 + \Delta x \cdot \frac{2\rho \cdot C_{ow}}{\sigma_{ow}})}, \frac{1}{-(2 + \Delta x^2 \cdot Q(x_1))}, \dots, \frac{1}{-(2 + \Delta x^2 \cdot Q(x_{N-1}))}, 1]^T$$

combining Equations (4.7), (4.9) and (4.23) yields

$$P(x_n, t) = (Y_n(t) - A_l[n] \cdot P(x_{n-1}, t) - A_u[n] \cdot P(x_{n+1}, t)) \cdot A_m[n] \quad (4.24)$$

Computation of $\underline{P}(t)$ implemented, running Jacobi relaxation for J iterations. Sabo et al. 39 assumes that the convergence tests will catch the cases in which the computation of $\underline{P}(t)$ does not converge after J iterations. The equation of iteration j is:

$$P^j(x_n, t) = (Y_n(t) - A_l[n] \cdot P^{j-1}(x_{n-1}, t) - A_u[n] \cdot P^{j-1}(x_{n+1}, t)) \cdot A_m[n]; 0 \leq n \leq N \quad (4.25)$$

\underline{A}_l , \underline{A}_m and \underline{A}_u are constant, solved in advance and stored on shared memory. Division by mass replaced by multiplication by $\frac{1}{M_n}$. The pressure $P(x_n, t)$ computed per cochlear section x_n on different thread. Hence each thread performs three multiplications, two add/sub operations and seven shared memory accesses for one Jacobi iteration. Inter-thread synchronization is required however after each Jacobi iteration since each thread uses the result of its neighbors for the next iteration. Since synchronization between threads that belong to the same block is much more efficient in CUDA Sabo et al. 39 limited N_x to the size of each CUDA block. While CUDA block can contain up to 1024 threads, main limitation for resources were shared memory available per block, therefore $N_x = 256$. This resolution proves to ensure convergence

of the algorithm and meet resource limitation demands on block size. The block starts solving the model for time $t = t_{BTI} \cdot (BL - 1)$.

4.3 Updating The computing algorithm to reduce errors

In Section 4.1.3 the signal pressure $P_{in}(t)$, is being used at the Y_0 initial condition due to the pressure transmitted trough the oval window Eq. (4.8) therefor we need to evaluate $P_{in}(t + \Delta t)$, since P_{in} is defined for

$$P_{in}(T_s \cdot k), k \in \mathbb{N} \quad (4.26)$$

To evaluate $P_{in}(t)$ for every time t which doesn't satisfy integer of t/T_s , a linear time interpolation was implemented Eq. (4.22).

This was implemented in code by using single precision variable to indicate time from the start of signal denoted as t and another indicator for the start of block time interval t_{bti} offset from the start will be denoted $nearest_s$. the formula for $\Delta\tau$,

$$nearest_s = \left\lfloor \frac{t - t_{bti}}{T_s} \right\rfloor \quad (4.27)$$

$$\Delta\tau = \frac{t - t_{bti} - nearest_s \cdot T_s}{T_s} \quad (4.28)$$

$\Delta\tau$ is single precision variables, all arithmetic binary floating point calculations on CUDA compatible devices are done by IEEE-754 2008 standard [34, Section G.2]. IEEE 754 floating point register is divided to 23 mantissa, 8 exponent and one sign bit. decimal precision is

$$d_{precision} = \frac{23 \cdot \log(2)}{\log(10)} = \frac{23}{\log_2 10} = 7.2 \quad (4.29)$$

if $\Delta\tau \leq 10^{-d_{precision}} \cdot t$ than

$$P_{in}(t + \Delta\tau) == P_{in}(t) \quad (4.30)$$

Sabo et al. 40 used 0.32 input signals, this means that for $t > 0.1seconds$ Eq. (4.30) for $\Delta\tau < 10^{-8}seconds$, convergence for this intervals is not guaranteed. 2 methods considered to solve this time interpolation domain

1. using double precision calculations - will lose 87.5% and 96.88% for throughput for compute capabilities 2.0 and 6.0 respectively. due to 4 units of add,multiply and fused add-multiply double precision for every 32 or 128 single precision respectively.
2. using single precision Eq. (4.22) representing time as combination of n_s and t_δ , therefore each time we need t we substitute $t = n_s \cdot T_s + t_\delta$. Time step was set maximum value $\Delta\tau = 10^{(-6)}seconds$, Section 4.1.3, with those parameters, algorithm ensures

$$t_\delta \leq T_s + \Delta\tau \quad (4.31)$$

with $\Delta\tau \ll T_s$ we claim that effectively $t_\delta \leq T_s$, [43] set $T_s = 50\mu sec$, linear approximation of $P_{in}(t)$ will be updated for

$$\Delta\tau > 10^{\lg_{10}(T_s) + d_{precision}} \approx 3 * 10^{-11} \quad (4.32)$$

regardless of t .

4.3.1 Error Measurements

Basilar Membrane Velocity The difference between outputs for old input interpolation method and ours can be shown for both the Membrane Velocity(BMV) and Nerve

Response. Fig. 4.3 shows difference between the peak envelopes of BM velocity at

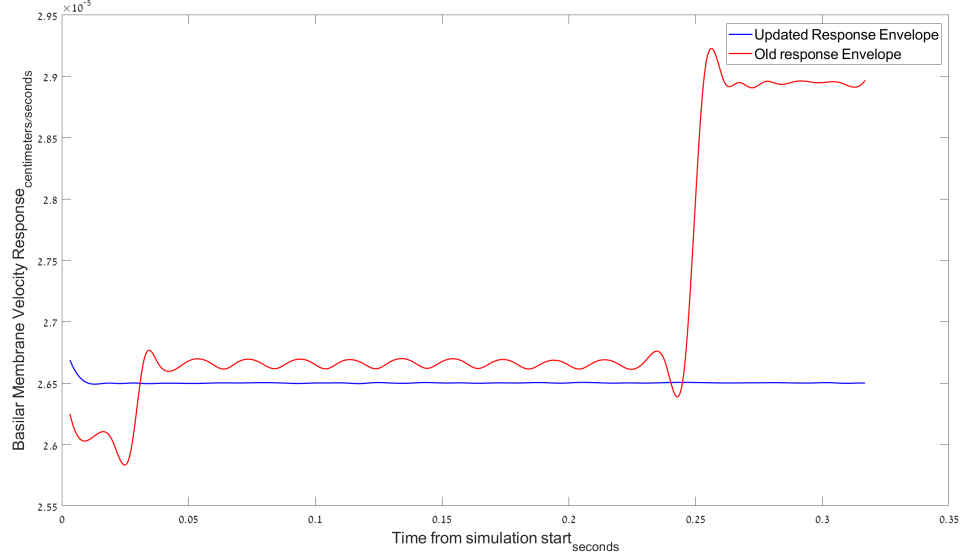


Figure 4.3: Envelope of Basilar Membrane velocity section in response for 4KHz tone 36dB with old(Red) and new(Blue) interpolation methods

section 0.64cm from OW for both the old and new algorithm. Response at relatively low power does not have feedback ripples, therefore an expected envelope needs to be approximately constant (except transition state). In the old interpolation method we see rippling, we also see peak jump at 0.025 and 0.25 seconds, since precision drop as direct function of $\log_{10}(t)$. HSR fibers excitation, λ_{high} measured in $\frac{\text{spikes}}{\text{second}}$, in response for 4KHz tone 40dB with old, Fig. 4.5, and new, Fig. 4.4, interpolation methods. At low Amplitude input and single tone signal, ANR expected to be cyclic (except transient stage).

As Sabo et al. 39, Ch 6 shown we need to ensure that our mismatches from the reference software are smaller than previous version. since this work intended to calculate Auditory Nerve Response difference will be measured both for Basilar Membrane Velocity and $\lambda_{high}(x, t)$ from Eq. (2.14). both velocity and auditory nerve errors will be

measured. For $\dot{\xi}_{bm}$ old and new interpolation errors will be tested and compared.

1. first defined average speed for membrane section with normalized input with power P

$$|\dot{\xi}_{bm}[i, P]| = \frac{1}{t_{end} - t_{start} + 1} \cdot \sum_{t=t_{start}}^{t_{end}} |\dot{\xi}_{bm}[i, P, t]|$$

defines signals average speed from index t_{start} to t_{end} at longitudinal section i .

2. to measure error for signals with range of amplitudes, an energy for the membrane measured on the reference program

$$E_{ReferenceMean}[P] = \sum_{i=0}^N |\dot{\xi}_{bm}[i, P]|_{referenceSoftware}^2$$

3. we then measure energy difference between the CUDA and reference program for both interpolation methods.

$$E_{InterpolationError}[P] = \sum_{i=0}^N [|\dot{\xi}_{bm}[i, P]|_{CUDA} - |\dot{\xi}_{bm}[i, P]|_{referenceSoftware}]^2$$

4. the normalized error will be function of input signal power P

$$Err[P] = \frac{E_{InterpolationError}[P]}{E_{ReferenceMean}[P]}$$

since we are interested to calculate JND from the results and this requires testing for multiple Pure Tone inputs, 3 frequencies chosen to examine effectiveness. with Oval Window self frequency of 1KHz and AC filter described at Table A.1, 500Hz, 1KHz, 2KHz, 4KHz examined for below cutoff and in AC range, on the Oval window cutoff and on slope of the AC filter, slightly after slope of AC filter and significantly DC response,

Auditory Nerve Response the AN errors measured against difference from λ_{spont} to ensure the rate will reflect JND measurements. we use HSR fibers as those are dominant when search for JND.

1. define average activity above static level for cochlea section i with signal power

P and interval $[t_{start}, t_{end}]$

$$\Delta\lambda_{AN}[i, P] = \frac{1}{t_{end} - t_{start} + 1} \cdot \sum_{t=t_{start}}^{t_{end}} (\lambda_{AN}[i, P, t] - \lambda_{spont})$$

2. creating reference software energy equivalent by square $\Delta\lambda$ along the cochlea.

$$E_{\lambda \text{ ReferenceMean}}[P] = \sum_{i=0}^N (\Delta\lambda_{AN}[i, P])^2_{referenceSoftware}$$

3. measuring equivalent error for cochlear sections for both the interpolation methods

$$E_{\lambda \text{ InterpolationError}}[P] = \sum_{i=0}^N [(\lambda_{AN}[i, P])_{CUDA} - (\lambda_{AN}[i, P])_{referenceSoftware}]^2$$

4. finally normalize error defined as

$$Err_{\lambda}[P] = \frac{E_{\lambda \text{ InterpolationError}}[P]}{E_{\lambda \text{ ReferenceMean}}[P]}$$

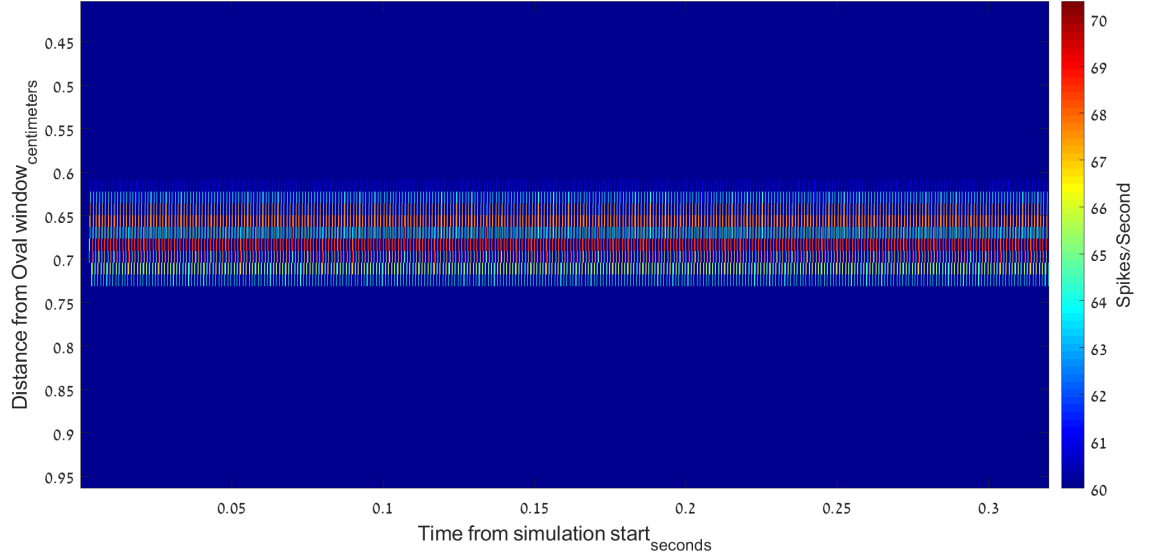


Figure 4.4: Result yielded by new interpolation method for $P_{in}(t)$, at 4KHz the nerves cant follow the membrane phase and gives constant yield which is proportional of $|P_{in}(t)|$ as observed experimentaly

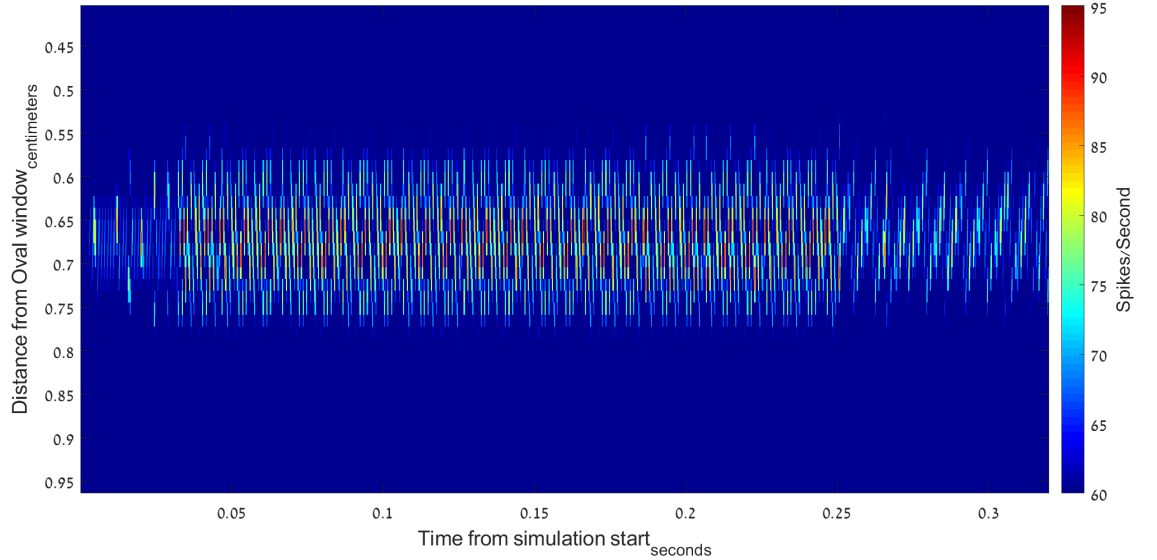


Figure 4.5: Result for old method of interpolation, we see nerves response change phase both temporarily and spatially in response for the degraded $d_{precision}$

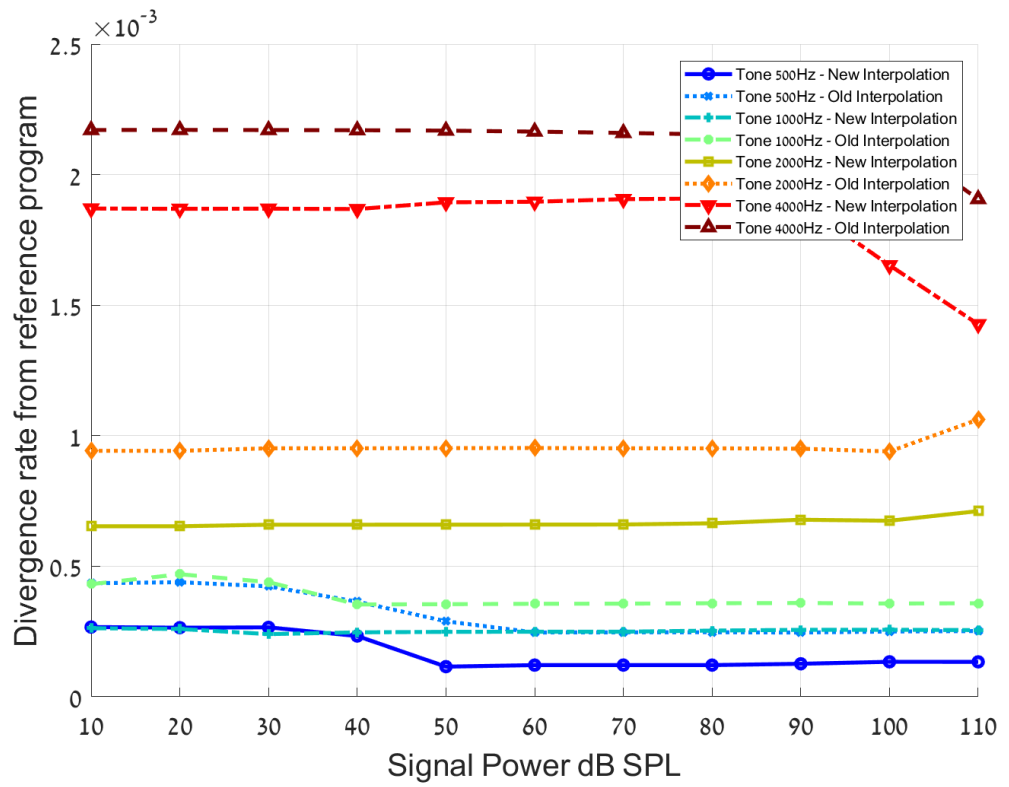


Figure 4.6: Error Mean difference show improvement for All frequencies and powers of relative error of $0.4 \cdot 10^{-3}$, for lower frequencies error is lowered for less then half of its former value

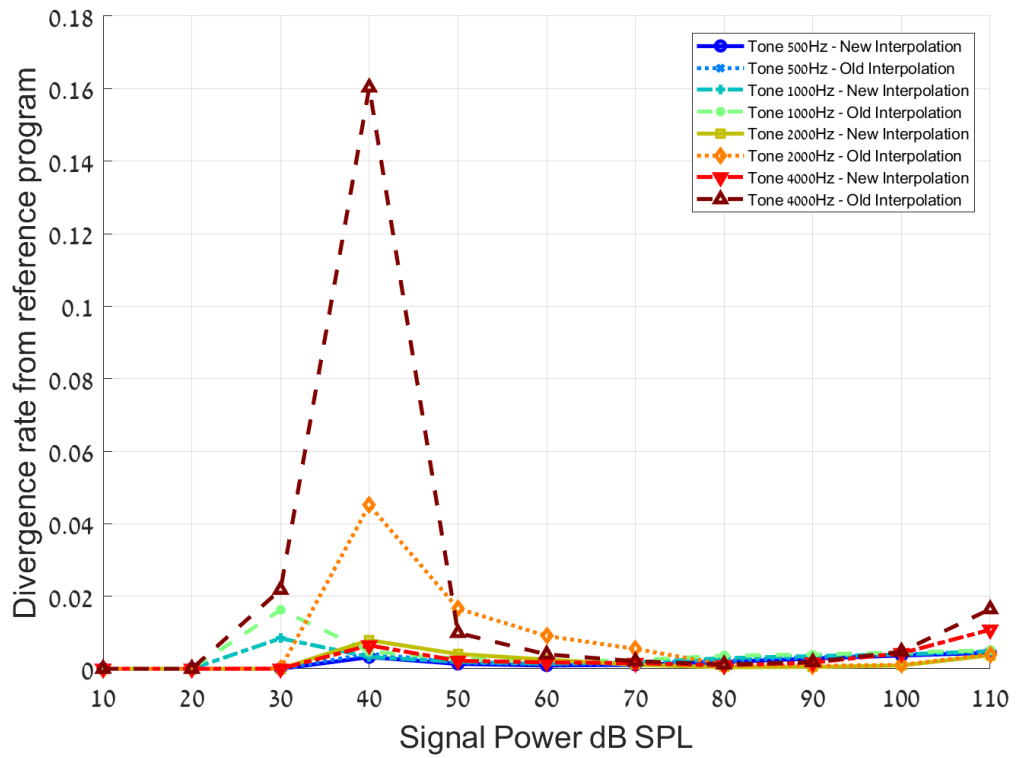


Figure 4.7: Error Mean difference show improvement for All frequencies, the effect however is most significant for 4KHz, error minimized from 16% to less than 1% for 40dB makes the new interpolation fit to calculate JND where the old version will err significantly

Chapter 5

Implementing Neural Response and JND evaluation

5.1 Neural Response Calculation

Here we will implement Massive parallel computing process to calculate Neural response from Section 4.2 results of Bassilar membrane velocity solution.

As shown at Section 2.2 Neural response can be described as homogeneous Poission when no signal is present but non homogeneous Poission (NHHP) response when signal is present. to calculate Neural response, IHC must be calculated first. parallelization of equation Eq. (2.12). h_{ihc} can be any linear filter, chosen parameters for tests described at Table A.1. this paralleled by first calculating IIR filter coefficients locally, and then solve on GPU, since IIR is recursive, parallelization is done over longitude dimension and over multiple power levels (JND section) but not on time domain, calculating is done by convolve the output for each section by the b and than convolve recursively the result with negative a coefficients.Implementation of the AC part from Eq. (2.12)

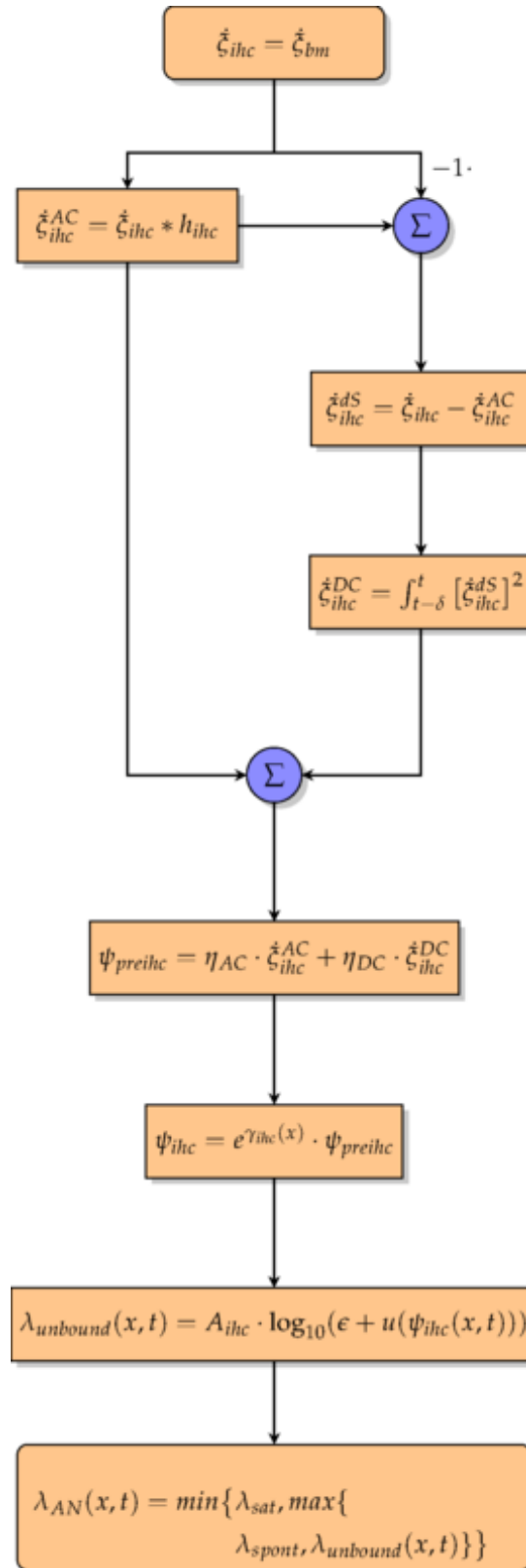


Figure 5.1: Flow chart of the Calculation of Auditory Nerve Response.

$$AC_{response}(x, t) = \sum_{i=0}^n (b(i) * BM_{velocity}(x, t - i)) - \sum_{i=1}^n (a(i) * AC_{response}(x, t - i)) \quad (5.1)$$

Denote n , number of coefficients in h_{ihc} , t is time index (results time is $t \cdot T_s$), x Cochlear Longitudinal Position, $BM_{velocity}$ is sampled Basilar membrane velocity, $\dot{\xi}_{bm}$ from Eq. (2.11) and $AC_{response}$ is the result. Program supports using FIR filters as well for this calculation, if FIR filter is chosen, paralleling will be done on time domain as well. We use the AC part of the IHC response to calculate the DC part, $\{\dot{\xi}_{ihc}(x, t) \cdot [1 - h_{ihc}(t)]\}^2$, by calculating intermediate result

$$dS_{high}(x, t) = \{BM_{velocity}(x, t) - AC_{response}(x, t)\}^2 \quad (5.2)$$

Equation (5.2) depends on results of Eqs. (2.11) and (5.1) at single time index, therefore can be done without synchronization. we approximate the DC part of the IHC voltage, $\int_{t-\delta}^t \{\dot{\xi}_{ihc}(x, t) \cdot [1 - h_{ihc}(t)]\}^2 dt$, from Eq. (2.12). this is DC response, by substitute dS_{high} in last equation and synchronizing before summary due to Eq. (5.3) relies on multiple time indexes.

$$DC_{response}(x, t) = \frac{1}{F_s * \delta} * \sum_{t_1=t-F_s*\delta}^t dS_{high}(x, t_1) \quad (5.3)$$

calculating $\log_{10}(\psi_{ihc}(x, t))$ as

$$V_{ihc}(x, t) = \eta_{AC} \cdot AC_{response}(x, t) + \eta_{DC} \cdot DC_{response}(x, t) \quad (5.4)$$

and substitute Eq. (5.4) in Eq. (2.12)

$$\log \psi_{ihc}(x, t) = \log_{10}(\psi_{ihc}(x, t)) = \log_{10}(\epsilon + \max\{0, (10^{\gamma_{ihc}(x)} \cdot V_{ihc})\}) \quad (5.5)$$

computation stages divided to ensure both data integrity (prevent reading result that not yet calculated) and maximum speed, since dependency on results on another time sample are present only in filters (calculating the $AC_{response}$ and Eq. (5.3) stages, all calculation that can be done on single time coordinate unified to single function such as the Eq. (5.2) result and Eq. (5.5) stage (implementing $\ln(1 + u(\psi_{ihc}(x, t)))$ from Eq. (2.14)), since lambda has 3 sets for the different types of neurons Section 2.2.1 as found at [43], A_{ihc} is dependent both on nerve type and spatial position, and solution for Eq. (2.14) is

$$\lambda_{AN}^H(x, t) = \min \left\{ \lambda_{sat}, \max \left\{ \lambda_{spont}^{(HSR)}, A_{ihc}^{(HSR)}(x) \cdot \log \psi_{ihc} \right\} \right\} \quad (5.6)$$

$$\lambda_{AN}^M(x, t) = \min \left\{ \lambda_{sat}, \max \left\{ \lambda_{spont}^{(MSR)}, A_{ihc}^{(MSR)}(x) \cdot \log \psi_{ihc} \right\} \right\} \quad (5.7)$$

$$\lambda_{AN}^L(x, t) = \min \left\{ \lambda_{sat}, \max \left\{ \lambda_{spont}^{(LSR)}, A_{ihc}^{(LSR)}(x) \cdot \log \psi_{ihc} \right\} \right\} \quad (5.8)$$

5.2 JND Calculation

Here we will implement JND calculation by methods of Rate Mean Square and All Information. Solving Eqs. (2.18) and (2.20)

where α is the signal amplitude, since both the signal+noise and noise only inputs are calculated together, JND computation will be ordered such that all dependent on another interval calculation will be solved first since CUDA cannot synchronize between blocks on the same kernel run [35]

5.2.1 Calculate Fisher Information

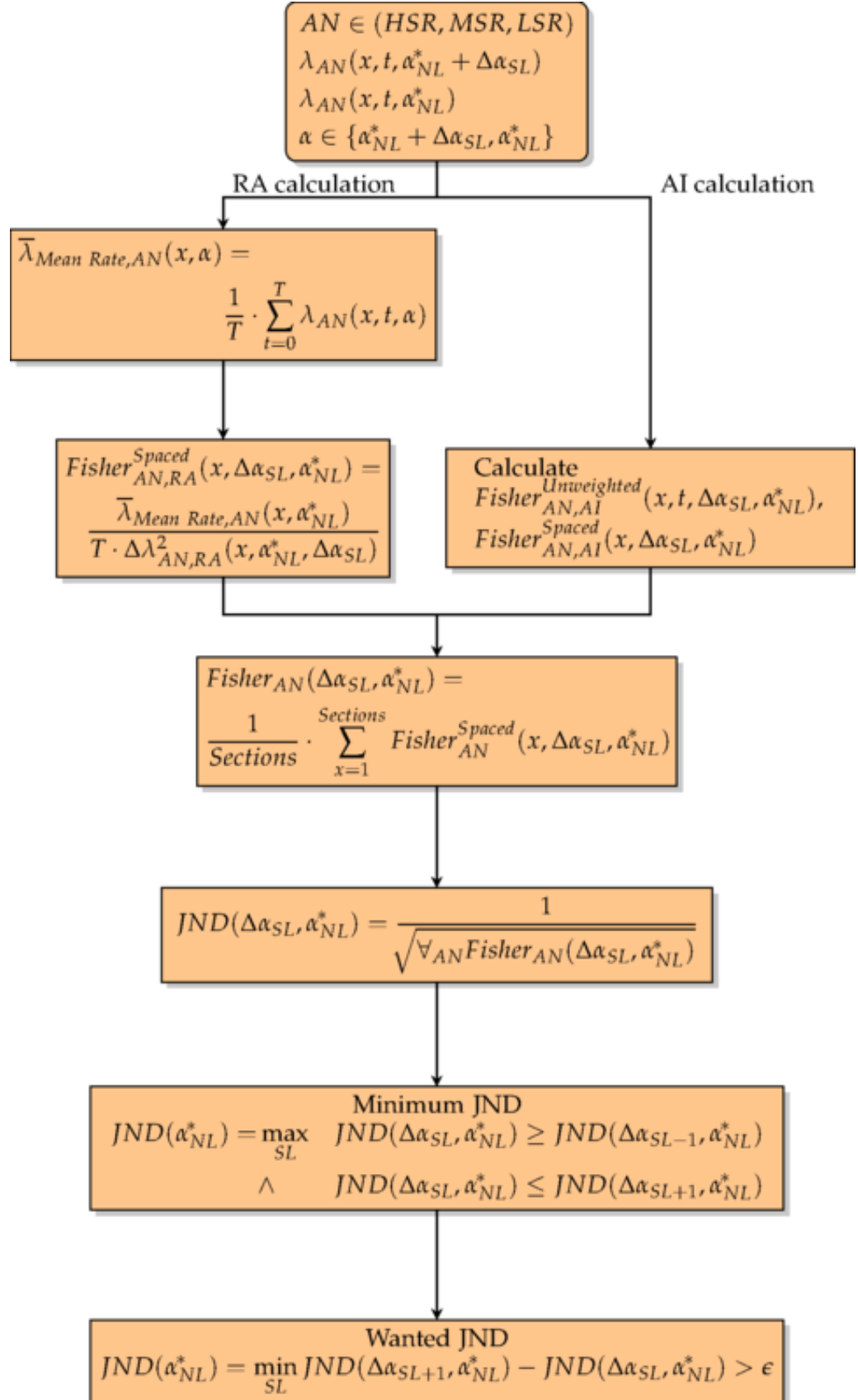


Figure 5.2: Flow chart of the Just Noticeable Difference Calculation.

Averaging Neural Response over time first stage will be to calculate for each signal and for every type of lambda

$$AN \in (HSR, MSR, LSR)$$

$$\alpha \in \{\alpha_{NL}^* + \Delta\alpha_{SL}, \alpha_{NL}^*\} \quad (5.9)$$

$$\bar{\lambda}_{AN,RMS}(x, \alpha) = \frac{1}{T} \cdot \sum_{t=0}^T \lambda_{AN}(x, t, \alpha)$$

since calculation happens in parallel, most stages will process data that was not result of other CUDA block on the JND calculation procedure.

However since Eq. (5.9) on RMS needs to subtract the averaging of the reference signal (silence or noise only) since each signal is calculate on separate CUDA block the averaging will be done in different kernel to synchronize the entire data [34].

From this point, all threads calculate process only their own data (or data processed on previous kernels).

Calculate Fisher formula for each Neural Group To evaluate CRLB in Eqs. (2.18) and (2.20) we need to substitute $\lambda(\alpha, x)$ with Eq. (5.9) result.

$$\Delta\lambda_{AN,RA}(x, \alpha_{NL}^*, \Delta\alpha_{SL}) = \frac{\bar{\lambda}_{AN,RMS}(x, \alpha_{NL}^* + \Delta\alpha_{SL}) - \bar{\lambda}_{AN,RMS}(x, \alpha_{NL}^*)}{\Delta\alpha_{SL}} \quad (5.10)$$

For AI will implement Eq. (2.27) for each longitudinal section x , by substituting α with α_{NL}^* and $\Delta\alpha$ with $\Delta\alpha_{SL}$ to get effect of signal over noise base response. Since ANR calculated on different kernel, data synchronization is guaranteed.

$$\Delta\lambda_{AN,AI}(x, t, \Delta\alpha_{SL}, \alpha_{NL}^*) = \frac{\lambda_{AN}(x, t, \alpha_{NL}^* + \Delta\alpha_{SL}) - \lambda_{AN}(x, t, \alpha_{NL}^*)}{\Delta\alpha_{SL}} \quad (5.11)$$

Each λ difference from same coordinate longitudinal and temporal from nerve response to noise only input (or silence), as noted, parallelization utilized to compute

response for different α in this case amplitude in parallel, so response is available.

than integrate Eq. (5.11) across temporal dimension

$$Fisher_{AN,AI}^{Unweighted}(x, t, \Delta\alpha_{SL}, \alpha_{NL}^*) = \frac{\Delta\lambda_{AN,AI}^2(x, t, \Delta\alpha_{SL}, \alpha_{NL}^*)}{\lambda_{AN}(x, t, \alpha_{NL}^*)} \quad (5.12)$$

We calculate Fisher formula for all information by first aggregate each longitudinal section Eq. (5.12) results over time to $Fisher^{Spaced}$

$$Fisher_{AN,AI}^{Spaced}(x, \Delta\alpha_{SL}, \alpha_{NL}^*) = \frac{M(x)}{T} \cdot \sum_{t=0}^T Fisher_{AN,AI}^{Unweighted}(x, t, \Delta\alpha_{SL}, \alpha_{NL}^*) \quad (5.13)$$

We substitute Eq. (5.9) in Eq. (2.18) to get the inverse square of $CRLB_{RA}$ per spatial section.

$$Fisher_{AN,RMS}^{Unweighted}(x, \Delta\alpha_{SL}, \alpha_{NL}^*) = \frac{\bar{\lambda}_{AN,RMS}(x, \alpha_{NL}^*)}{T \cdot \Delta\lambda_{AN,RMS}^2(x, \alpha_{NL}^*, \Delta\alpha_{SL})} \quad (5.14)$$

We than substitute Eq. (5.14) in Eq. (2.18) and multiply by number of nerves per section to get $CRLB_{RA}$.

$$Fisher_{AN,RA}^{Spaced}(x, \Delta\alpha_{SL}, \alpha_{NL}^*) = \frac{M(x)}{Fisher_{AN,RMS}^{Unweighted}(x, \Delta\alpha_{SL}, \alpha_{NL}^*)} \quad (5.15)$$

CRLB use nerve density function $M(x)$ as tuned by [43], to calculate Fisher on both methods (Rate Mean Square and All Information) will average across longitudinal dimension

Aggregate Fisher Information over space

$$Fisher_{AN}(\Delta\alpha_{SL}, \alpha_{NL}^*) = \frac{1}{Sections} \cdot \sum_{x=1}^{Sections} Fisher_{AN}^{Spaced}(x, \Delta\alpha_{SL}, \alpha_{NL}^*) \quad (5.16)$$

since $Sections = 256$ we parallelize the process by using the modified parallel reduction algorithm 5.1, at this point we have $signals \cdot AN$ which is few hundreds to few thousands points.

```

unsigned int tid=threadIdx.x;
unsigned int i=blockIdx.x*(blockDim.x*2)+threadIdx.x;
sm[tid] = d[i]+d[i+blockDim.x];
__syncthreads();
for (stride=blockDim.x/2;stride>=1;stride>>=1)
{
    if (tid<stride) sm[tid]+=sm[tid+stride];
    __syncthreads();
}

```

Algorithm 5.1: CUDA implementation of parallel reduction algorithm page 113 of [24]

Aggregate Fisher Information over neural groups since last stage require to summarize over AN use of GPU is unnecessary since run time is small fracture of percents from the program, optimizing it will not give us noticeable speeding factor, from this stage forward calculation done identically for both Rate Mean Square and All Information, solving Eq. (2.25), this will be done as it seen

$$JND(\Delta\alpha_{SL}, \alpha_{NL}^*) = \frac{1}{\sqrt{\sum_{\forall AN \in (HSR, MSR, LSR)} Fisher_{AN}(\Delta\alpha_{SL}, \alpha_{NL}^*)}} \quad (5.17)$$

Calculate Just Noticeable Difference per noise level since JND depends on α , it is necessary to find an optimal alpha, to do this we will use gradient decent, the JND is low barrier and as [43], show, 2 possible methods for optimal alpha, calculation is done from the signal processed multiple times in parallel as shown in Section 6.1

1. Minimum JND - solve by finding $\alpha^* + \Delta\alpha_{level}$ such that

$$\begin{aligned}
 JND(\alpha_{NL}^*) = \max_{SL} \quad & JND(\Delta\alpha_{SL}, \alpha_{NL}^*) \geq JND(\Delta\alpha_{SL-1}, \alpha_{NL}^*) \\
 \wedge \quad & JND(\Delta\alpha_{SL}, \alpha_{NL}^*) \leq JND(\Delta\alpha_{SL+1}, \alpha_{NL}^*)
 \end{aligned}$$

2. Wanted JND - solve by finding $\alpha^* + \Delta\alpha_{\text{level}}$ such that

$$JND(\alpha^* + \Delta\alpha_{\text{level}+1}) - JND(\alpha^* + \Delta\alpha_{\text{level}}) > \epsilon(\Delta\alpha)$$

with ϵ decided as function of $\Delta\alpha$, we approximately solving Eq. (2.27)

Chapter 6

Optimizations

6.1 Parallel Input Generation

6.1.1 Requirements

To calculate JND of signal either pure tone or recorded signal as show at Section 5.2.1.

We need to measure JND for multiple $\Delta\alpha_{\text{level}}$, we define set of power instances in dB

$\Delta\alpha_0, \Delta\alpha_1, \dots, \Delta\alpha_{\text{level}}$ with

$$\Delta\alpha_{\text{level}} = \Delta\alpha_0 + SL \cdot dBJump \quad (6.1)$$

The series of $\Delta\alpha_{\text{level}}$ can be decided at input level allow search for $JND(\alpha^*)$ at different resolutions. testing can be done on

- pure tone - pure tone is cosine signal and can be defined by its frequency, its amplitude (noted as SL - signal level) and length
- recorded signal - can be any signal

since α^* is noise amplitude,Eq. (2.27) requires Auditory Nerve response to $s(t, \alpha^*)$, we implemented 3 types of noise

- Quiet - $n_{UF}(t) = 0$ for every $0 \leq t \leq F_s \cdot T$

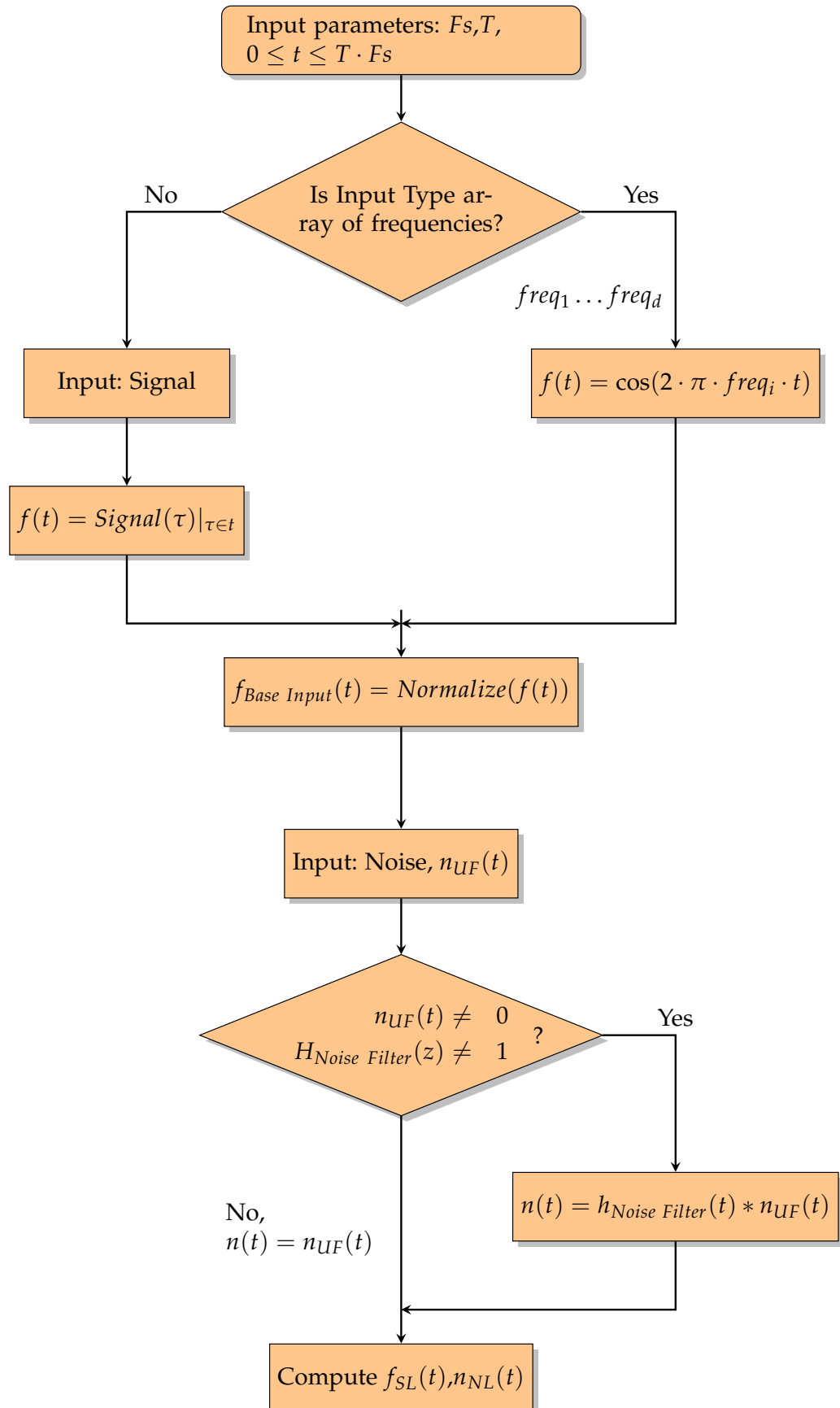


Figure 6.1: Flow Chart of Input generation. Continues at Fig. 6.2

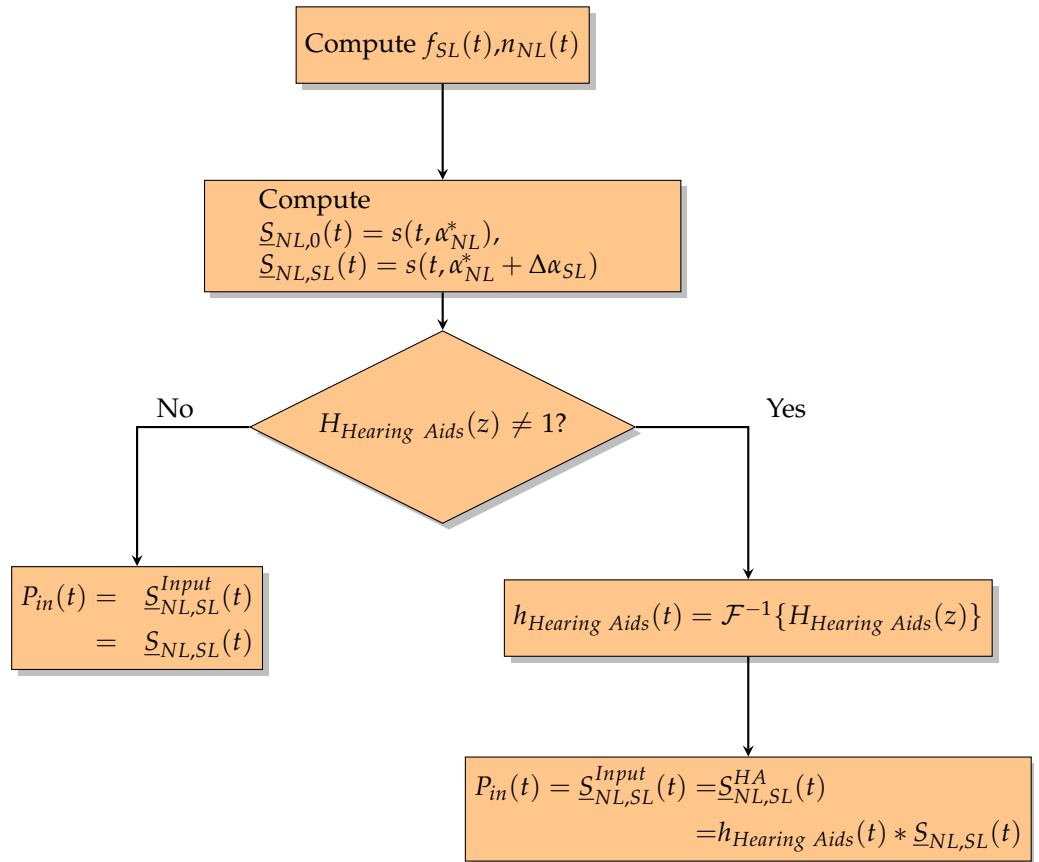


Figure 6.2: Flow Chart of Input generation continues from Fig. 6.1

- White Noise - $n_{UF}(t) = \mathcal{N}(0, 1)$ for every $0 \leq t \leq F_s \cdot T$ UF is unfiltered
- Recorded noise - can be any noise at length of $M \leq F_s \cdot T$, if $M < F_s \cdot T$ than $n_{UF}(t) = 0, M \leq t \leq F_s \cdot T$
- Filtered - takes white noise or recorded signal and pass them through linear filter

if filter noise isn't used $n(t) = n_{UF}(t)$ to solve Eq. (2.27) for array of Noise Levels amplitudes, NL, gains $\alpha_1^* \dots \alpha_{NL}^*$ therefore we wish to calculate IR to \underline{S}

$$\underline{S} = \begin{Bmatrix} s(t, \alpha_1^* + \Delta\alpha_0) & \dots & s(t, \alpha_1^* + \Delta\alpha_{SL}) & s(t, \alpha_1^*) \\ \vdots & \ddots & \vdots & \vdots \\ s(t, \alpha_{NL}^* + \Delta\alpha_0) & \dots & s(t, \alpha_{NL}^* + \Delta\alpha_{SL}) & s(t, \alpha_{NL}^*) \end{Bmatrix} \quad (6.2)$$

6.1.2 Normalization

as shown by [43] multiple Signal Pressure Level (SPL) reference levels needed to be tested in order to find numerical approximation to much experimental results at [42].

if noise filtered is present,

$$H_{Noise\ Filter}(z) = \frac{\sum_{j=0}^{m_1} b_j \cdot z^{-j}}{\sum_{k=0}^{m_2} a_k \cdot z^{-k}}$$

$$h_{Noise\ Filter}(t) = \mathcal{F}^{-1}\{H_{Noise\ Filter}(z)\}$$

$$n(t) = h_{Noise\ Filter}(t) * n_{UF}(t)$$

different normalization methods are tested for signal f or noise n , denoted as χ with time length of T . process will be defined as $\chi_{Base\ Input}(t) = NormalizeInput(\chi(t))$

- normalize by power, getting $\frac{1}{T} \cdot \int_0^T \chi^2(t) dt$ with $\chi(t)$ is the un normalized signal pressure over time the input function is

$$\chi_{Base\ Input}(t) = \frac{1}{\left(\frac{1}{T} \cdot \int_0^T \chi^2(t) dt\right)} \cdot \chi(t)$$

- normalize by energy, to testing various signals at same strength, division will be done on summary of signal energy.

$$\chi_{Base\ Input}(t) = \frac{1}{\left(\int_0^T \chi^2(t)dt\right)} \cdot \chi(t)$$

- skip normalization for experiments values decided outside the program

$$\chi_{Base\ Input}(t) = \chi(t)$$

- normalize for power of different length, since signal period effects JND, this allows to test if short signals with period T_s with same energy as long signals, period T_l will have same JND.

$$\chi_{Base\ Input}(t) = \frac{1}{\left(\frac{1}{T} \cdot \int_0^T \chi^2(t)dt\right)} \cdot \chi(t)$$

as shown at Section 5.2.1, we need to calculate signal for range of $\Delta\alpha_{level}$ hence the equation is

$$f_{SL}(t) = SPL_{ref} \cdot 10^{\frac{\Delta\alpha_{level}}{20}} \cdot f_{Base\ Input}(t) \quad (6.3)$$

$$n_{NL}(t) = SPL_{ref} \cdot 10^{\frac{\alpha_{NL}^*}{20}} \cdot n_{Base\ Input}(t) \quad (6.4)$$

$$s(t, \alpha_{NL}^*) = n_{NL}(t) \quad (6.5)$$

$$s(t, \alpha_{NL}^* + \Delta\alpha_{SL}) = n_{NL}(t) + f_{SL}(t) \quad (6.6)$$

SPL_{ref} , Table A.1, is the pressure level at 0dB SPL.

6.2 Optimizing For Kepler Architecture

Our project was developed from Sabo et al. 40 work, which was implemented on GTX590, CC 2.0, its architecture includes 32K registers per SM Section 3.1, to prevent Tail End effect described at Achieved Occupancy. BM velocity calculation kernel

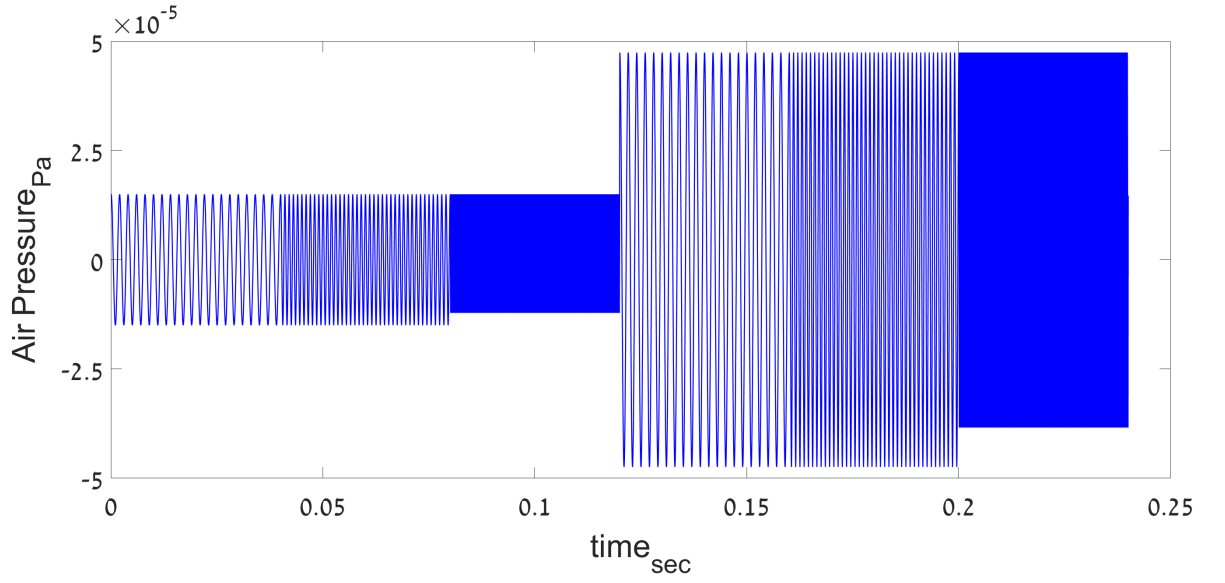


Figure 6.3: Example for Eq. (6.2) implementation. Generated Input for $\Delta\alpha_{SL}$ as 40dB and 50dB SPL and α_{NL}^* of 0 (Noiseless), Pure tones at frequencies of 0.5, 1 and 4 K Hz.

limited to 2 blocks per SM, this gives 512 threads with 63 registers, 31.5K per SM, full capacity. improvements factors for each configuration shown at Fig. 6.5 and registers needed per thread at Fig. 6.4.

M1 Kepler, CC 3.0 has 64K registers, twice the amount. this will allow 4 blocks to run on SM without register spilling as shown by Section 3.2. Kepler use the L1 Cache for temporary register spilling [34, G.4.1] and configurable between cache and SHM. optimizing for cache require reducing SHM use to 16kB per SM, increasing occupancy to 5 blocks allows 3.2KB. original program required 11KB shared memory, even under 4 blocks.

M2 Cache would have 3KB per block and would have problem to contain temporary register spills. examining Eq. (4.23), \underline{A} and $P(X_n, t)$, by denoting $P(x_N, t) = 0$ boundary condition Eq. (4.9), we can change all values of \underline{A}_u to 1 and ignore

it completely, this saves 1028B. the \underline{A}_l can be replaced by formula $n < N - 1$, reduce another 1028B of SHM. the kernel needs to synchronize memory fence 3 arrays in SHM, P to calculate $P^{(j)}$ from $P^{(j-1)}$, $|f(y(t + \Delta t), t + \Delta t) - f(y(t), t)|$ from Eq. (4.19) to calculate maximum value and third parameter can be seen at Oded and Furst 36(Pg. 37)

$$e = \sum_{x_n} \|\ddot{\xi}_{bm}^{(L)}(x_n, t_{n+1}) - \ddot{\xi}_{bm}^{(L-1)}(x_n, t_{n+1})\|_2 \quad (6.7)$$

to calculate summary across dimension x . the remain SHM representing $\ddot{\xi}^{(L)}$, $\ddot{\xi}^{(L)}$, P_{TM} , K_{bm} from Table A.1 are replaced by registers. last parameter stored $\frac{L \cdot \Delta t}{2}$ need to test between threads is that $\max_{x_N} \frac{L \cdot \Delta t}{2} > 1$, Section 4.1.2 replaced by register and using `--syncthreads_or` allow to synchronize test between the block threads without using SHM. while this increase register pressure to 62 per thread, effect is minimal due to requirement of copying the SHM to register before using the value. this also reduce usage of SHM to less than 3.2KB per block allow us 5 blocks.

M3 Many of the operations require arithmetics of the same constants, Kepler Architecture allow fast access to Constant memory Chapter 3 with 64KB, this allow us to replace most of those calculations with constants, thus both reduce unnecessary arithmetics and lower register pressure to 58. the calculation replaced are α_s / α_l , Δx^2 , $\Delta x \cdot \sigma_{ow} \cdot \gamma_{ow}$. we also remove previous tests of linearity that were set already and added unnecessary commands to the code.

M4 we note that Oval window parameter to calculate (Eq. (4.7), Y_0) require $\ddot{\xi}_{OW}^{(L)}$, $\ddot{\xi}_{OW}^{(L)}$, $\ddot{\xi}_{OW}^{(L)}$, $\ddot{\xi}_{OW}^{(L-1)}$, $\ddot{\xi}_{OW}^{(L-1)}$, $\ddot{\xi}_{OW}^{(L-1)}$. this calculations applied only to thread 0 which han-

dele the oval window boundary condition calculation and takes only 24 Bytes,CUDA compiler optimizing registers allocation to satisfy executive branch, if needed in another registers will be used for different functions on separate branches. this reduce registers used per thread to 54.

M5 CUDA fused multiply-add command allows both operations without passing through the registers.by fusing all possible multiply-adds this reduce registers used per thread to 51.

M6 by setting launch bounds maximum number of blocks to 5 the compiler attempts to reduce register pressure to maximum that allows 5 blocks occupancy, 48 in this case. the other 6 spilled to L1. occupancy is $256 \cdot 5 = 1280$ threads which take 7680 Bytes in L1 by increasing capacity of L1 to 48KB we avoid round trips to L2 due Cache miss. 5 blocks occupancy gives on configuration of 7 frequencies, 10 levels of power ($\Delta\alpha$),3 noise levels, 40 m sec for each input (20 m sec for each block interval) + reference for each noise $3 * (10 * 7 + 1) * 2 = 426$ Blocks intervals. GTX 760 has 9 SM, handling improves from 36 to 45 at once. run time change from 6.08 to 5 seconds, 21.5% improvements.

6.3 Optimizing For Pascal Architecture

Main GPU Structure changed between Kepler & Pascal/Maxwell Fig. 3.1. optimizations for higher occupancy did not give us faster algorithm. due to register spilling, on Pascal devices, it stored on L2 Cache as oppose to L1 on Kepler. we have tested multiple algorithms to compare several convergence methods and division, the large

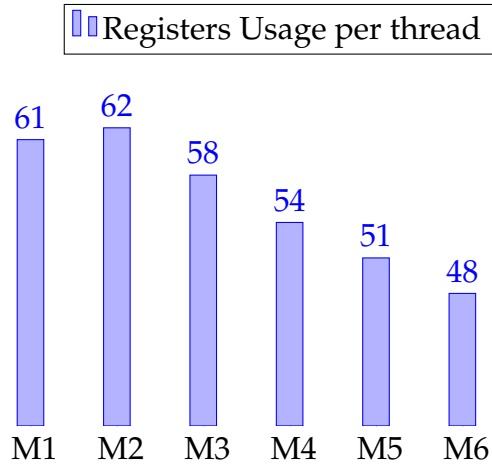


Figure 6.4: Comparison of Kepler Register Per Thread Requirements for implementations M1 - M6 from Section 6.2

L2 cache allows to compile with launch bounds of 6 blocks per SM, the register are spilling to cache. we use GTX 1080 Ti with 28 SM, we setup a run of 4 noise power levels(none,10,15 and 20dB),25 signal's (0-72dB by jumps of 3) on 7 different frequencies with each interval set to 40ms, thats 1400 blocks on the entire card, this gives 10 blocks per SM on 5 blocks. we compared the 6 methods factor of speed(normalized) and get for the first 6 optimizations only M2 gives significant acceleration of 66%. CUDA 8

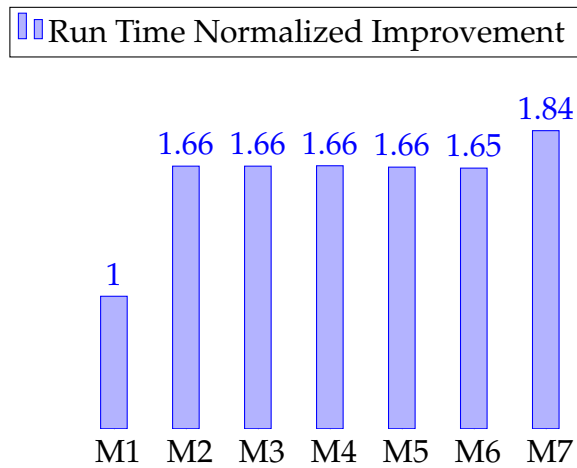


Figure 6.5: Comparison of Pascal Run times for different configurations, each number indicates $\frac{runtime(M1)}{runtime(Configuration)}$. for example, if M1 to 3.52 seconds M2 will take $\frac{3.52}{1.66} = 2$ seconds

compiler changed to standard of C++ from C. we combined template functions with

warp shuffle to reduce synchronizations. as noted at M2, we require 3 aggregations to test (maximum and 2 additions). we use modified warpReduceSum algorithm with template to get 3 accumulator operate on each value, with block of 256 we reduced to 8 mid values for each function, we run single thread synchronization and take 3 values for each of the lower threads and run second partial summary on the remained 8. this gives another run time reduction by 10% to total improvement of run time by 84%.

6.4 Testing Congestion With Nsight

To test critical bottlenecks when optimizing at 6.3 ,for M5,M6 and M7. several criteria from Section 3.2 tested include Instruction Statistics, Issue Efficiency and total run time for the kernel as measured by Nsight, since architecture counters are limited Nsight configuration measured for noiseless input at 16 - 20 levels of signals (depends on the maximum occupancy of the configuration) and very short interval of 8ms. we verified against run time for standard run of 4 noise levels (none, 33, 43 and 53dB) for 7 frequencies and 10 levels of $\delta\alpha$ (40 ms), M8 is M6 with synchronization method of M7

while Nsight is not indicative for run time of the kernel, it can be used to analyze stall issues, for cases without the triple synchronization mechanism (M5,M6). synchronization is stalling issue 38% against 27% when mechanism added.

6.5 Optimizing Convergence Parameters

The program optimized to generate large Database of mapping damage (In Outer and Inner Hair cells) profile to JND, optimal upper bounds to speeds measured in Lipschitz criteria Eq. (4.19) searched by detect, run time of BM velocity kernel for 100 damage

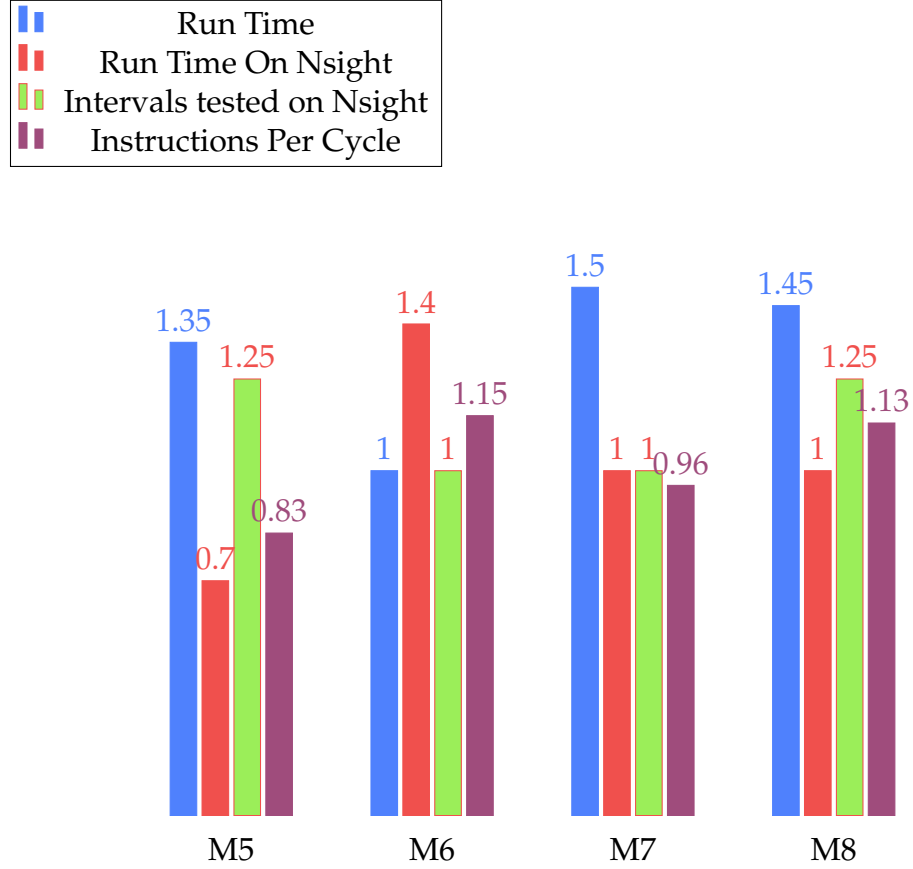


Figure 6.6: Comparison of run time on Nsight as debug mode and on Matlab as run time, Intervals amount tested, M5-8 described at Sections 6.2 and 6.3. Instructions Per cycle can be maximally 8 if there are 4 warps that can issue 2 instructions every cycle. this is not the case here, as stall issues block warps around 75% of the clock cycles. It is also noticeable that Nsight run time is negatively predict tested run time

profile (cartesian product of OHC and IHC by 10% jumps). each profile tested with 4 level of noise. silent, 33,43 and 53dB

Testing show non significant error increase when increasing minimal Δt for the algorithm from order of 10^{-15} to 10^{-8} and speed increase of nearly 30%.

6.6 Optimize JND calculation for single tones

Examination of the Eq. (2.18) shows us that CRLB depends on T linearly, if we can prove that for large enough T $\bar{\lambda}$ independent of T for pure tone input. we can show by numeric evaluation that after initial conditions pass $t > T_{start}$ for every input with

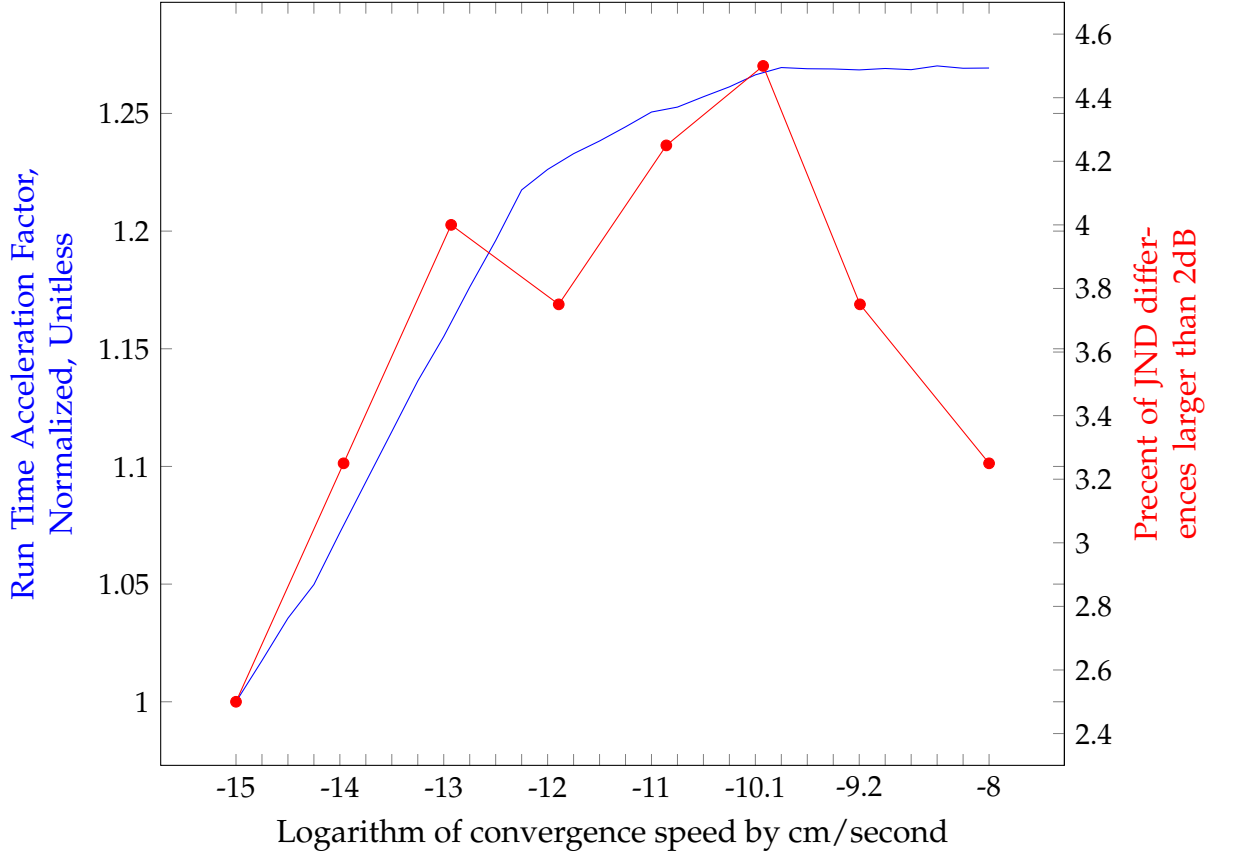


Figure 6.7: Testing JND accuracy and algorithm acceleration as function of minimum Δt in Lipschitz criteria. We tested 2 criteria. First run time acceleration factor over base algorithm run time at convergence speed of $10^{-15}cm$. Second is percent of JND measurements differs more than $2dB$ from reference algorithm of Oded and Furst 36 to avoid large differences.

form of S_{input} , and with frequency which is multiplication of $250Hz$. since, it has been shown [6, 36] that WKB linear approximation over gain by $10-20dB$ and become unreliable near the ω_{CF} due to backward propagation. we will test numerically, optimal interval will be short as possible but will vary minimally from similar interval, we denote the main Int with length of T , composed of concatenated k_{JND} (Eq. (6.8a)) (in case of k_{JND} not integer, we round down) intervals each of length Eq. (6.8a)

$$T_{shortened} = \frac{T}{k_{JND}} \quad (6.8a)$$

$$Int = [I_1 \dots I_{k_{JND}}] \quad (6.8b)$$

we then process for each interval sine function of length $t_{BOP} + T_{shortened}$ with tested part of $T_{shortened}$. instead of $t_{BOP} + T$ and replace Equation (2.18) with

$$CRLB_{substituted}(\alpha^*) = \left\{ \frac{k \cdot T_{shortened}}{\bar{\lambda}(\alpha^*)} \left[\frac{\partial \bar{\lambda}(\alpha)}{\partial \alpha} \Big|_{\alpha=\alpha^*} \right]^2 \right\}^{-\frac{1}{2}} \quad (6.9)$$

By substituting Eqs. (6.8a) and (6.8b) in

$$\bar{\lambda}(\alpha) = \frac{1}{T_{shortened}} \cdot \int_{t_{BOP}}^{t_{BOP}+T_{shortened}} \lambda(t, \alpha) dt \quad (6.10)$$

covariance of I_j from I_k calculated as

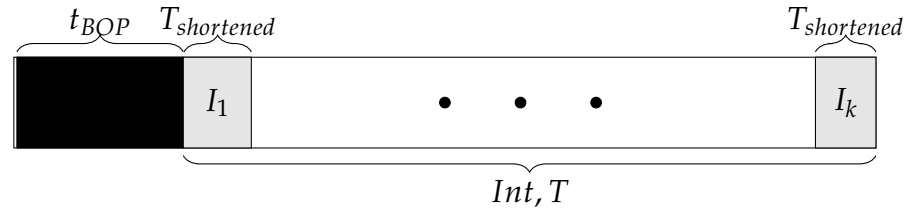


Figure 6.8: Division of Interval to measure JND, $T_{shortened}$ from Eq. (6.8a) and Int from Eq. (6.8b) and t_{BOP} described at Fig. 4.2

$$T_{offset}(j) = Tr + (j - 1) \cdot T_{shortened} \quad (6.11a)$$

$$I_j - I_l = \lambda(x, t + T_{offset}(j), \alpha) - \lambda(x, t + T_{offset}(l), \alpha) \quad (6.11b)$$

$$var(I_j, \alpha) = \sqrt{\int_0^{T_{shortened}} \int_0^{L_{co}} [\lambda(x, t + T_{offset}(j), \alpha)]^2 dt dx} \quad (6.11c)$$

$$= \sqrt{\sum_{t=0}^{T_{shortened}/T_s \text{ Sections}-1} \sum_{x=0}^{T_s \text{ Sections}-1} [\lambda[x, t + T_{offset}(j), \alpha]]^2 dt dx}$$

$$covar(I_j, I_k, \alpha) = \frac{var(I_j - I_k, \alpha)}{var(I_k, \alpha)} \quad (6.11d)$$

$$covar_{avg}(I_j, I_k, \alpha) = \frac{1}{k-1} \sum_{m=1}^{k-1} covar(I_m, I_k, \alpha) \quad (6.11e)$$

We use Eq. (6.9) with Eq. (2.18) to find the bound

$$|CRLB(\alpha^*) - CRLB_{substituted}(\alpha^*)| \leq covar_{avg}(I_1, I_k, \alpha^*) \quad (6.12)$$

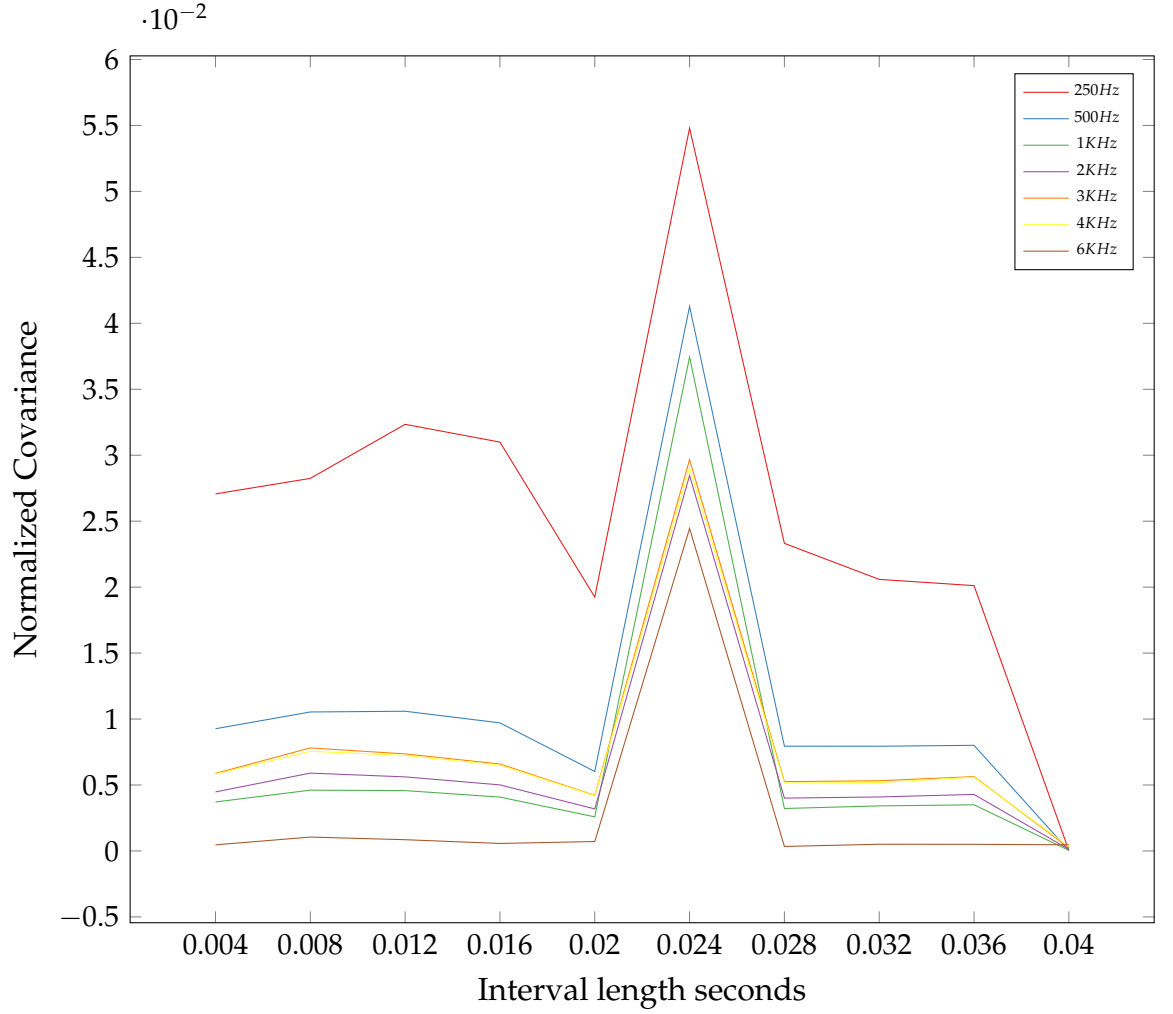


Figure 6.9: Testing JND accuracy as function of interval length, X axis is tested interval length in seconds, Y axis is normalized covariance from Eq. (6.11e), each plot measures for different frequency tone, described at legend

we now can test multiple $T_{shortened}$ values to find optimal value.

From Fig. 6.9 optimal $T_{shortened}$ is 20ms instead of 0.2seconds, acceleration by factor

of $\frac{200ms + T_r}{20ms + T_r} = \frac{212ms}{32ms}$, 6.625.

6.7 Optimizing Execution Flow

One of our main goals is to improve run time for BM velocity, ANR and JND calculation we also need to present results afterwards. We do it graphically using Matlab. Sabo et al. 39 used to launch *dos* [26] command that would run program from the Op-

erating System. This methodology has several drawbacks which slows the program, We upgrade the flow for Matlab to load *DLL* from *.mex* file. The process described in Figs. 6.10 and 6.11, it is high level description of the flow which for box contains / the text before it describe Sabo at al version and the text after describes ours. The stages in light orange are identical for both versions, while the older program didnot contained ANR or JND calculation, at this level of abstraction they are identical to BM velocity flow. Identical stages are of 2 kinds, first is upload data to GPU Read Access Memory, RAM, and download it to the CPU RAM. The second kind is the calculation themselves, done on the GPU.

1. First change in the flow described in the purple boxes. executing the *dos* command cause the operating system to open new process for each run, involving the scheduler [52], creates stack and heap for the program, at the end of the execution process those resources are released. our implementation shares memory space with Matlab, that means that we do not need to allocate and release those resources for each run, Matlab preserve the loaded function in memory after the first run. this saves *0.7seconds* each run regardless of the Input/Output size.
2. Second change involves Input/Output, older program and Matlab did not shared memory space, their only option to communicate was trough HD read and write which is slow. the differences are in the green boxes. Older method required reading data from multiple files, albeit small ones and write data, including large files, ANR and BM velocity output takes together *80MB* per 1 second of input at *20KHz* sampling rate. typical execution can process up to 20 seconds of Sound

at one time, if we want to see the results our program will need to write 1GB to the disk and Matlab read same amount of data, this takes 2 seconds, rest of the programs took only 4. Reading and writing to the HD should be avoided. Our program shares memory space with Matlab and can copy results through C Matrix Api [25]. memory copy of 1GB to matlab array took less than 50ms, made it 40 times faster.

3. Third change is result of memory allocation and deallocation. When the program executed from Operating System it needs to acquire and release its own memory, else it will become inaccessible. Arrays are allocated both on GPU for computation and on CPU for the output this takes approximately 20 seconds for 1.25 GB. our program allocates memory only when current buffers are not large enough or program runs for the first time, which usually happens few times in normal mode of operations (creating large database involves running thousands time on same size input)

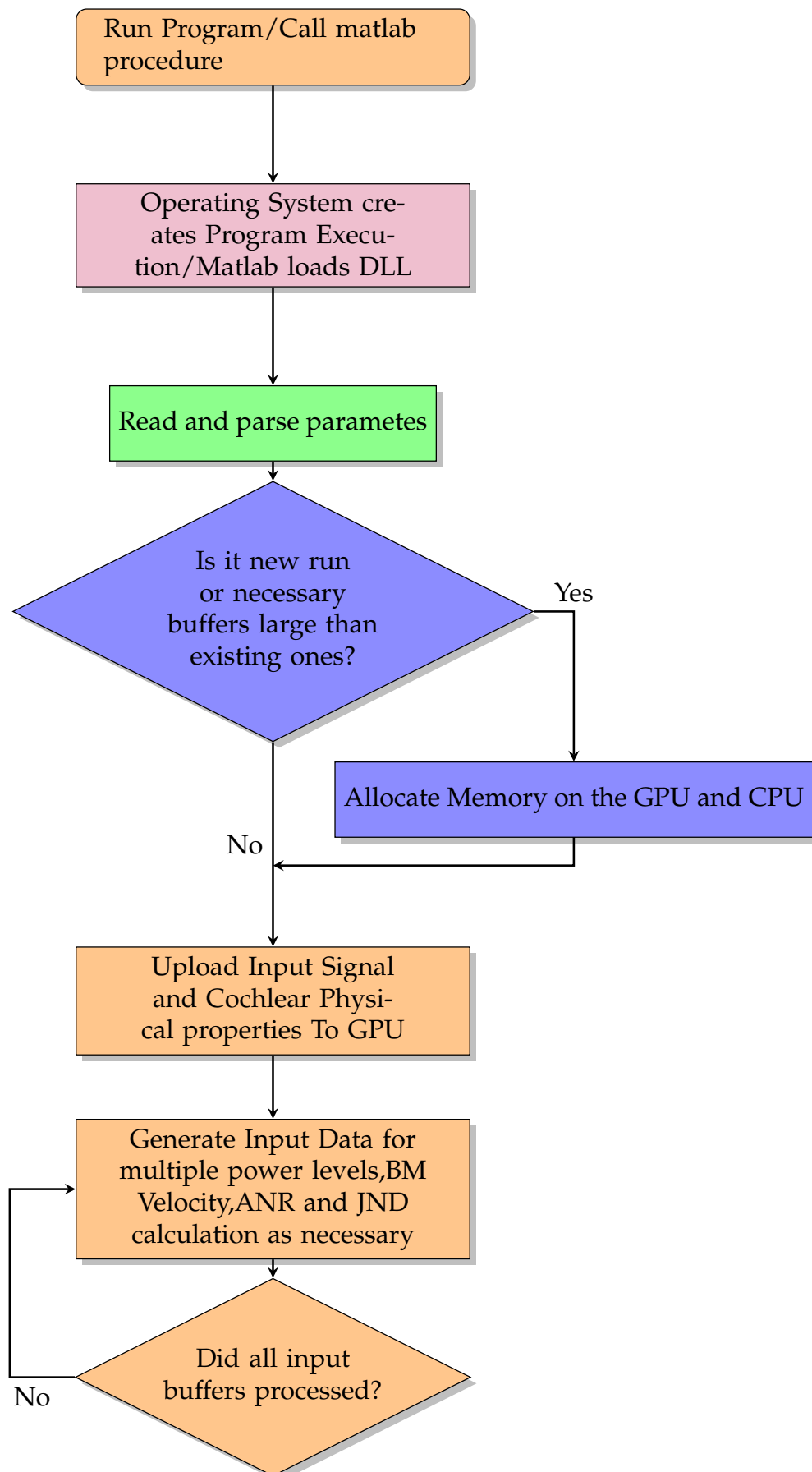


Figure 6.10: Flow chart of program input and Cochlear solver

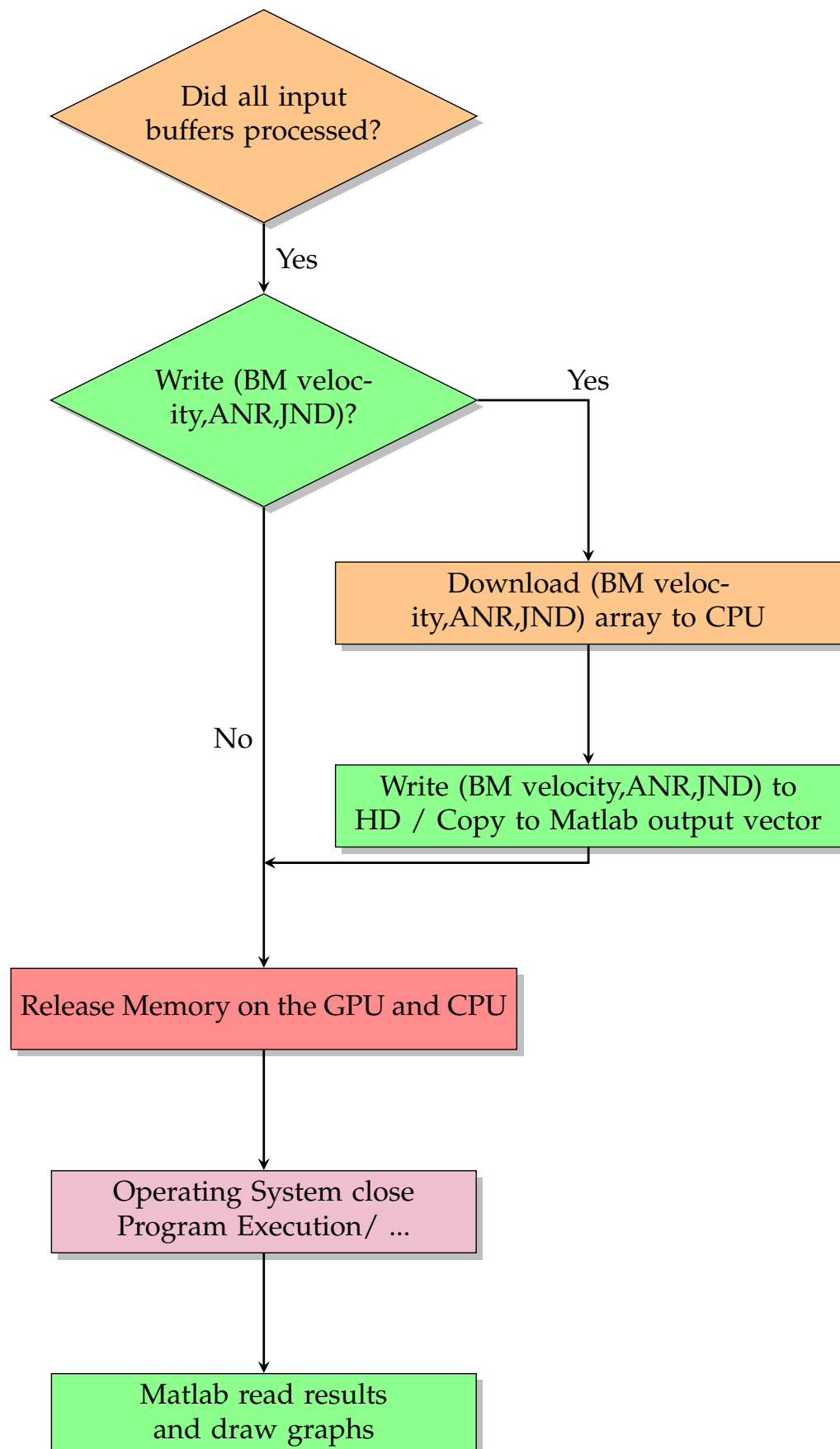


Figure 6.11: Flow chart of program. Output and release memory

Chapter 7

Hearing Aids Effectiveness

7.1 Characteristics

Hearing impairments happens in multiple forms but the most common among them is the sensory neural hearing loss [10]. difficulties in speech understanding come from reduce audibility on some of the frequencies and complete loss on others. this cause the processed waveform for the patient to be different from original signal. the most critical frequencies for speech understanding are between 500Hz and 4KHz. since most energy of the speech is in bands below 500Hz, but most information is above it loosing audibility in the range cause speech to be detectable but not understood. leading to statements about lack of clarity and mumbling of the speakers to hearing impaired. for correcting this situation, aids must amplify different frequencies according to specific loss at the band. Gain-frequency response defines full profile of prescription for the patient. if amplification is dependent on frequency alone but not amplitude, its linear hearing aid. this chapter we will demonstrate the effectiveness of program to analyze patient damage profile and different aids prescriptions.

7.2 Hearing Aid History

while hearing amplification methods were used since ancient times, such hand to the ear, until the 20th century all method were acoustical. this methods had several limitations, they were large due to necessity of having large open end to collect maximum possible acoustic energy and long tube to gradually concentrate the collected sound. a too shorter tube will cause most of the pressure waves to reflect back. the lack of active amplifier also means that amplification cannot be above collected energy, due to energy preservation, gain-frequency profile is also limited, the device cant be adjusted to amplify specific band reliably. at 1902 the first commercial electronic hearing aid become available. those are carbon based aids assembled from microphone with diaphragm that pressed dust or granules which functioned as variable resistor that control current inside the receiver, this system amplified sound by 20-30dB, with additional carbon amplifier, 70dB gain is achieved, while gain-frequency response profile can be set by arrange multiple systems in parallel with different, dust concentrations, those were limited by their frequency resolution and the noise the dust itself created limiting understandability. the vacuum tube (VT) had much better amplification profile than the carbon dust but were too large and energy consuming to be practical as mobile aids until the 1940's. with transistor become commercial in 1952, all VT hearing aids replaced with transistor aids in 1954. the transistor has better amplification characteristics than the VT, it also does not need heating up and is smaller than VT, which enable manufacturers to reduce of the device and create the first Behind The Ear(BTE) aid. invention of the Integrated Circuit(IC) allow to put multiple transistors

in single device. since 1980's reduction in transistor size and use of FET meaning that the signals can be processed numerically and use of digital control, users can adjust their own setting by switch between multiple amplification profiles for different situations. reduction in size leads to aids completely inside the ear canal make them hidden, according to the desires of the patients to avoid the stigma of hearing loss.

Audiology as scientific research started in 1930's. originally amplifications adapted to mirror the Audiogram HL, however this cause to over amplification at high amplitudes, causing discomforts and saturations for the amplifier makes speech unintelligible. at 1944 Lynbarger set the half gain principle, this means that gain for each frequency will be half of the Audiogram HL

7.3 Hearing Aid, Gain Calculations

7.3.1 Hearing Aids Simulation

The program can simulate Linear Hearing Aids effects for patient, with FIR and IIR transfer functions. if hearing aid is present we will use the \underline{S}^{HA} instead of \underline{S} (Eq. (6.2)), to define \underline{S}^{HA} , hearing aid frequency domain transfer function without loss of generality can be

$$H_{HearingAids}(z) = \frac{\sum_{j=0}^n b_j \cdot z^{-j}}{\sum_{k=0}^m a_k \cdot z^{-k}}$$

with n and m as the number of coefficients in nominator and denominator accordingly.

we use in time domain as $h_{Hearing Aid}(t) = \mathcal{F}^{-1}\{H_{Hearing Aids}(z)\}$,

$$P_{in}(t) = \underline{S}_{NL,SL}^{HA}(t) = h_{Hearing Aid}(t) * \underline{S}_{NL,SL}(t)$$

7.3.2 Database Generation

Our program allowed Starwitsky and Furst 43 to generate large database of more than 10000 different impairment profiles. to test hearing aid prescriptions for the patient we will scan the database and take the OHC/IHC profile that satisfy

$$\min_{\substack{\text{OHC/IHC} \\ \text{profile}}} \sqrt{\sum_{f \in (\text{Measured Frequencies})} (JND_{\text{Patient}}(f) - JND_{\text{SPL}}^{\text{profile}}(f))^2} \quad (7.1)$$

we denote $JND_{\text{Patient}}(f)$ as the measured JND SPL at frequency f and $JND_{\text{SPL}}^{\text{profile}}(f)$ as function that get frequency f for profile and return JND SPL in dB, value loaded from the database. The program can emulate hearing aids formula as part of the cochlea 6.1, thus allow to calculate JND with the hearing aids effect (α is calculated before passing the signal trough hearing aid). we will use formulas based on [11], those give values only for specific frequencies. to find full gain profile, we use linear interpolation for the rest frequencies. we define NH as normal hearing such that $JND_{\text{SPL}}^{NH}(f)$ return JND for healthy model at frequency f . the JND HL will therefore can be substituted

$$JND_{\text{HL}}^{\text{profile}}(f) = JND_{\text{SPL}}^{\text{profile}}(f) - JND_{\text{SPL}}^{NH}(f) \quad (7.2)$$

a prescription method is function that get JND and returns gain. denoted as $gain_{\text{prescription}}$.

our gain method will be

$$\text{DiscreteGain}(\text{profile}, \text{prescription}, f) = \text{gain}_{\text{prescription}}(JND_{\text{HL}}^{\text{profile}}(f)) \quad (7.3)$$

this however allow only use for known frequencies from [43] work. we denote series

$f_1 \cdots f_n$ that at speech bandwidth will satisfy $f_{i+1} - f_i \gg 20\text{Hz}$. we denote $F_{\text{nyq}} = \frac{F_s}{2}$

such that

$$f_i^{\text{norm}} = \frac{f_i}{F_{\text{nyq}}} \quad (7.4)$$

and $f_{-1}^{norm} = 0, f_{n+1}^{norm} = 1$. we can now define average gain for each bandwidth G_i^{BW} , each defined between range of $\frac{f_{i-1}^{norm} + f_i^{norm}}{2} + \epsilon_f \dots \frac{f_i^{norm} + f_{i+1}^{norm}}{2}$. with $\epsilon_f = \frac{20Hz}{F_{nyq}}$. we will define dense grid of frequencies with interval Δf such that $\frac{1}{\Delta f} \gg n$ and number of points in grid are $n_{pt} = \frac{1}{\Delta f}$. the gain for interpolated point at index k that satisfy $f_{i-1}^{norm} < k \cdot \Delta f < f_i^{norm}$ will be

$$BW_k^{position} = \frac{(k \cdot \Delta f - f_{i-1}^{norm})}{f_i^{norm} - f_{i-1}^{norm}}$$

$$MidGain_k = BW_k^{position} \cdot JND_{HL}^{profile}(f_i^{norm})$$

$$+ (1 - BW_k^{position}) \cdot JND_{HL}^{profile}(f_{i-1}^{norm})$$
(7.5)

the filter will than time shift by half length, to ensure real time domain filter, pre windowed H will be inverse FFT from gain concatenated to its mirrored conjugate, Inversed and than truncated to n

$$Gain_k = MidGain \cdot e^{-0.5 \cdot i \cdot \pi \cdot k \cdot \Delta f}$$

$$cGain_k = conj(Gain_k)$$
(7.6)

$$H_{pre\ window} = [Gain_1 \dots Gain_{n_{pt}} cGain_{n_{pt}-1} \dots cGain_2]$$

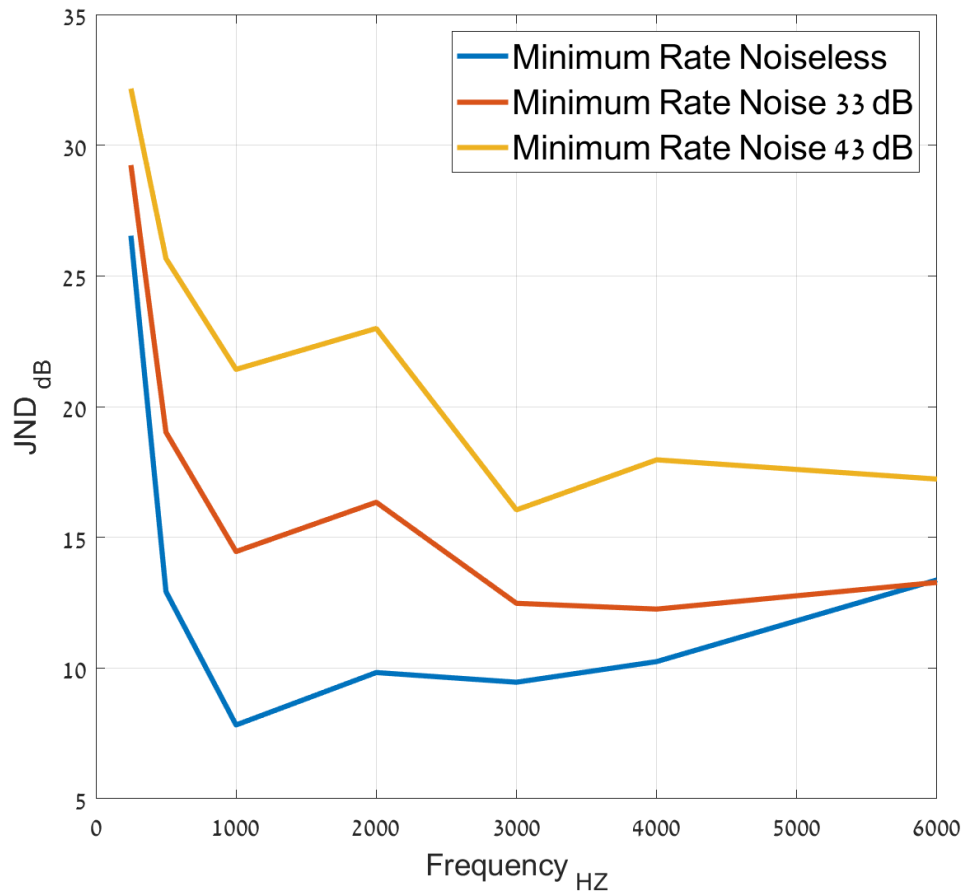
$$h_{pre\ window}(t) = iFFT\{H_{pre\ window}\}$$

calculating the hearing aid response filter h done by applying the window, tests show Kaiser to fit.

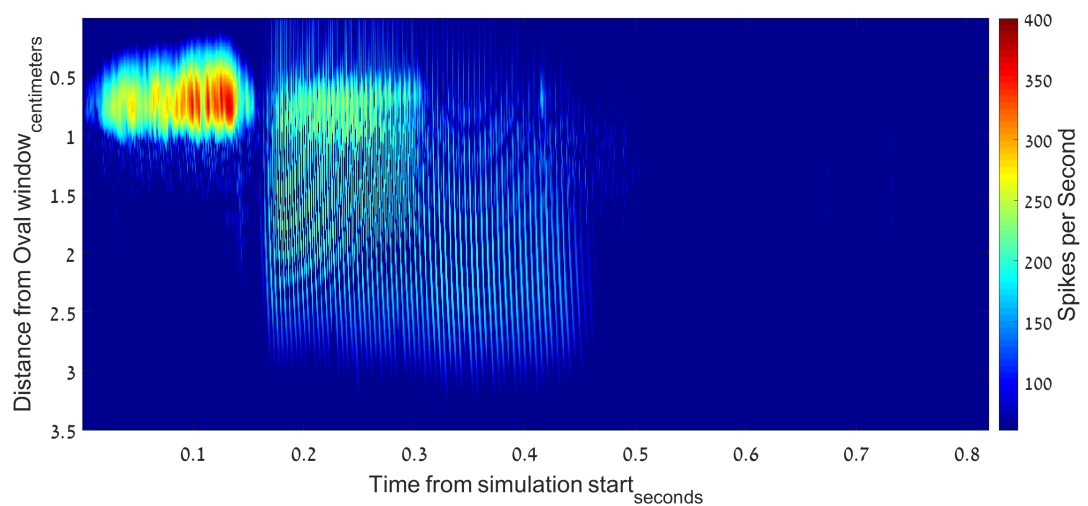
$$h(k) = h_{pre\ window}(k) \cdot \frac{I_0\left(\beta \cdot \sqrt{1 - \left(\frac{k - \frac{n}{2}}{\frac{n}{2}}\right)^2}\right)}{I_0(\beta)}$$
(7.7)

7.4 Hearing Aid, OHC damage

the patients detected profiles here based on the work of [43], using this program to much volunteers audiograms to OHC and IHC profiles. simulating noise will use Fig. 7.3 to emulate crowd noise.

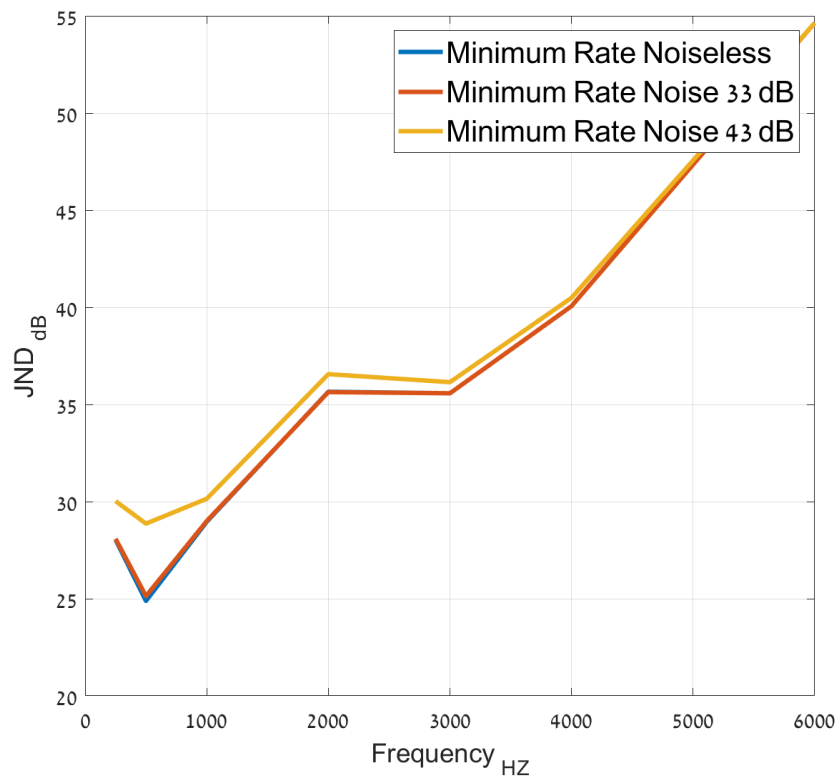


(a) Normal Hearing JND

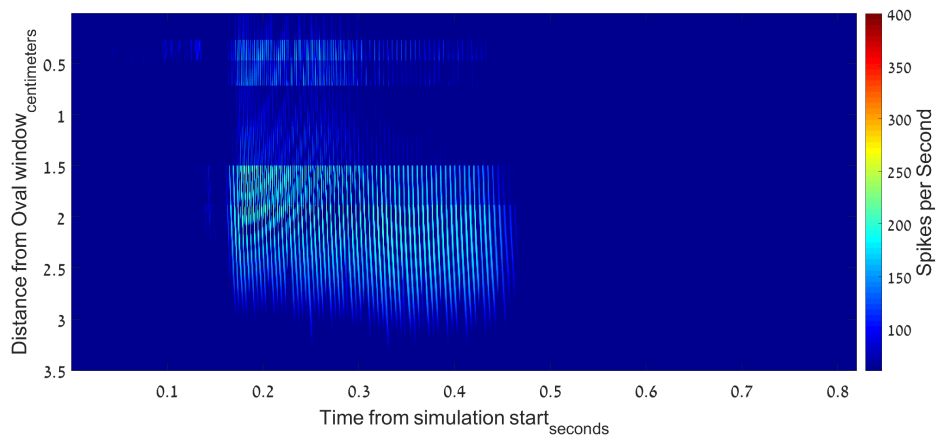


(b) Normal Hearing ANR

Figure 7.1: Normal Hearing model Auditory Nerve(ANR) response for the Hebrew word SHEN spoken by female at comfortable sound level, we see differences of detected signals, Simulation results close to real experiments. Detection of consonant SH shown on the neural response graph



(a) OHC at 25% and IHC at 87% JND



(b) OHC at 25% and IHC at 87% ANR

Figure 7.2: Patient with OHC at 25% and IHC at 87% ANR response for the Hebrew word SHEN spoken by female at comfortable sound level, we see differences of detected signals, comparing ANR and JND to healthy mode Fig. 7.1. We see for hearing impaired patient most consonants disappeared, the remain sound is weak HE, make the word difficult if not impossible to understand, also note that with IHC loss is not constant the IHC's pass 2.5-3KHz frequencies are damaged substantially more. this is also can be seen by the JND SPL graphs, using crowd noise passed through Butter-worth LPF with passband of 800Hz, spectrogram at Figure 7.3, stop-band of 1.2KHz with attenuations $-3dB$ and $-30dB$ respectively

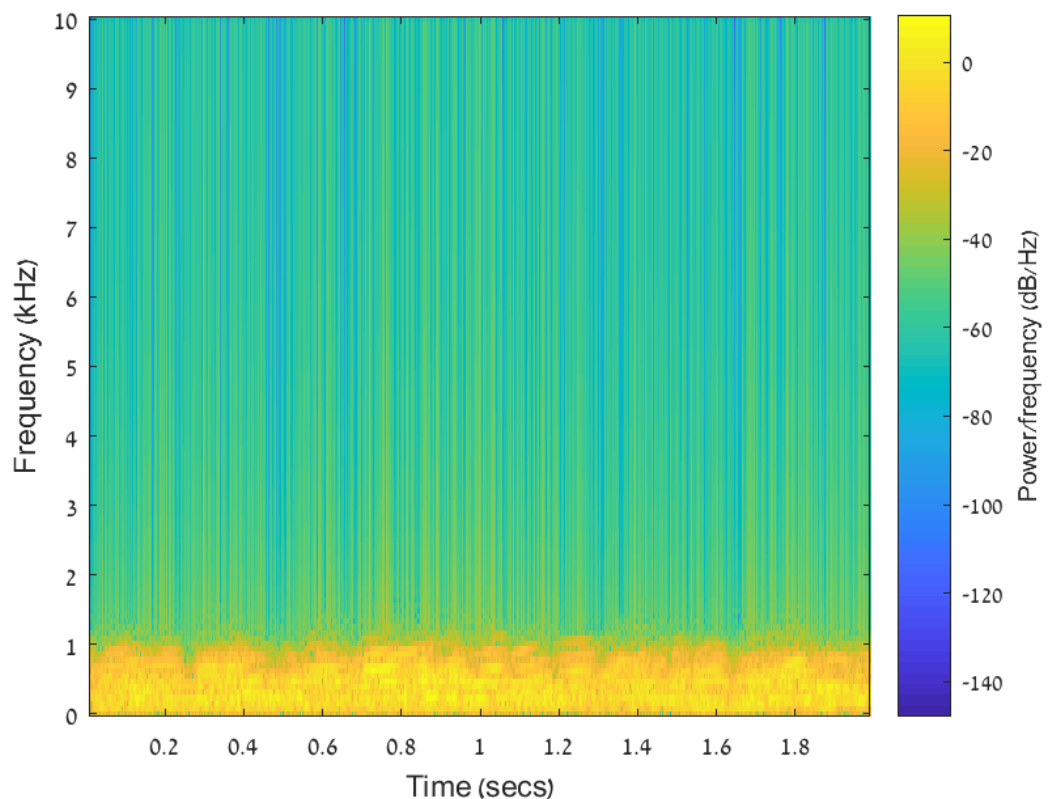


Figure 7.3: show normalized energy density of noise for JND test at Figure 7.1(a) and Figure 7.2(a)

first example at Figs. 7.1 and 7.2 shows comparison between hearing impaired and normal model. OHC damage cause mostly reducing hearing sensitivity for signals above 1KHz, the OHC characteristic frequency.

Measuring effects of different hearing aid prescriptions As denoted in Section 7.2, most of linear hearing aids follows half gain rule with some minor adjustments for 2 factors. first is that while most of the speech's energy is in the lower frequencies, less than 500Hz. most of the information is at 2-4KHz, second factor is that since most of high frequency sound pressure is absorbed by relatively small objects, most of the noise is lower frequencies, combine this factors with upward masking of the hearing

impaired means its preferable to amplify speech band (2-4KHz) much more than lower frequencies. 3 types of linear hearing aids tested

NAL Revised Method the NAL procedure (Dylon & Bryn 1986) is working to amplify each frequency to preferred hearing level, maximizing speech intelligibility. the formula takes summary of Audiogram HL values for 0.5, 1 and 2KHz and divide it by 20, it then add 0.31 of the audiogram HL of amplified frequency plus fix factor that amplify frequencies between 1-1.5KHz, lower amplification for frequencies below 1KHz significantly, and slightly lower others, therefore contribute to prevent upward masking from low frequencies.

Gain and Output (POGO) method is based on Lynbarger half gain principle with mild attenuations for lower bands, which contains most of the noise. this ensures easy to calculate method and improve speech intelligibility. POGO also includes formula to determine Maximum Power Output (MPO) by averaging user uncomfortable thresholds at 0.5, 1 and 2KHz. MPO was not implemented in our program due to this research focus on the lower hearing thresholds.

Berger method, this linear approach based on the assumptions that comfortable speech intensity is between 55 and 70dB with critical tones at 2-4KHz. as oppose to other methods the amplification is stronger than the half gain principle and approach the $\frac{2}{3}$. since band lower than 500Hz does not contribute for intelligibility, this frequencies won't be amplified in berger method.

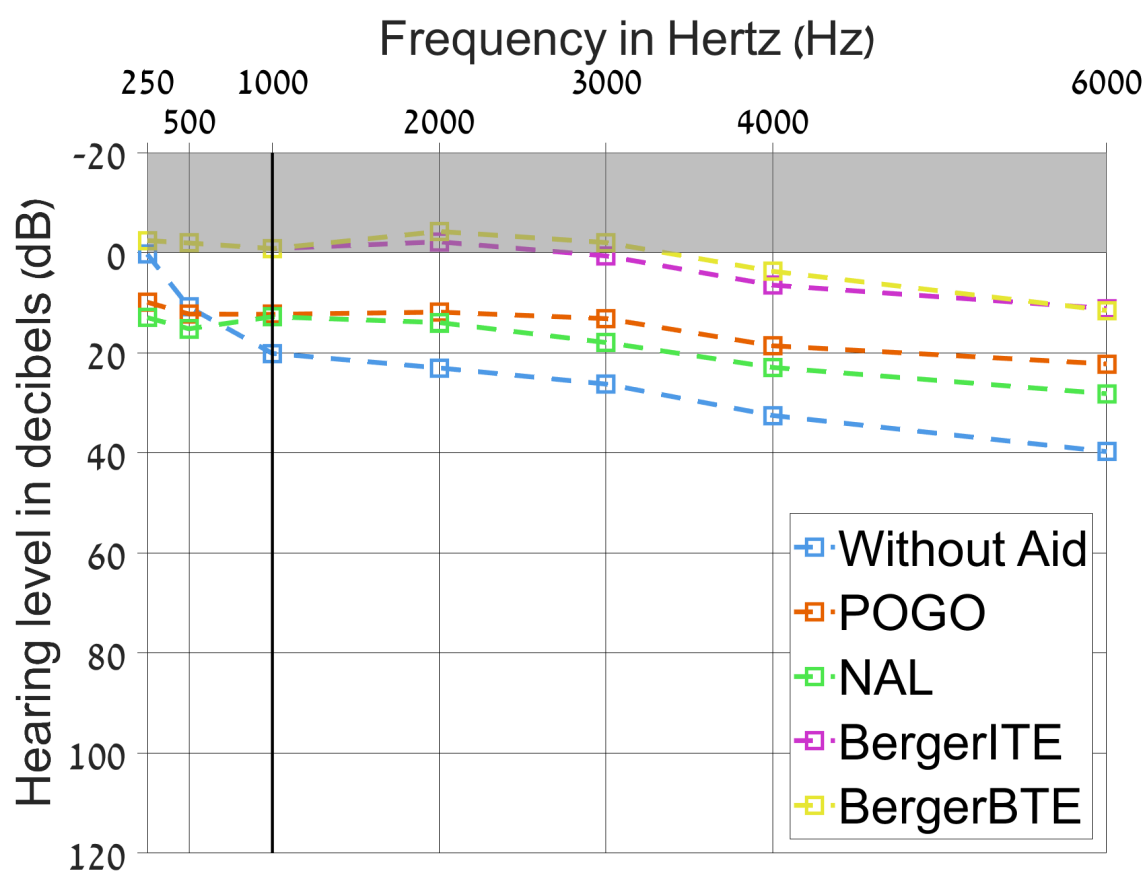
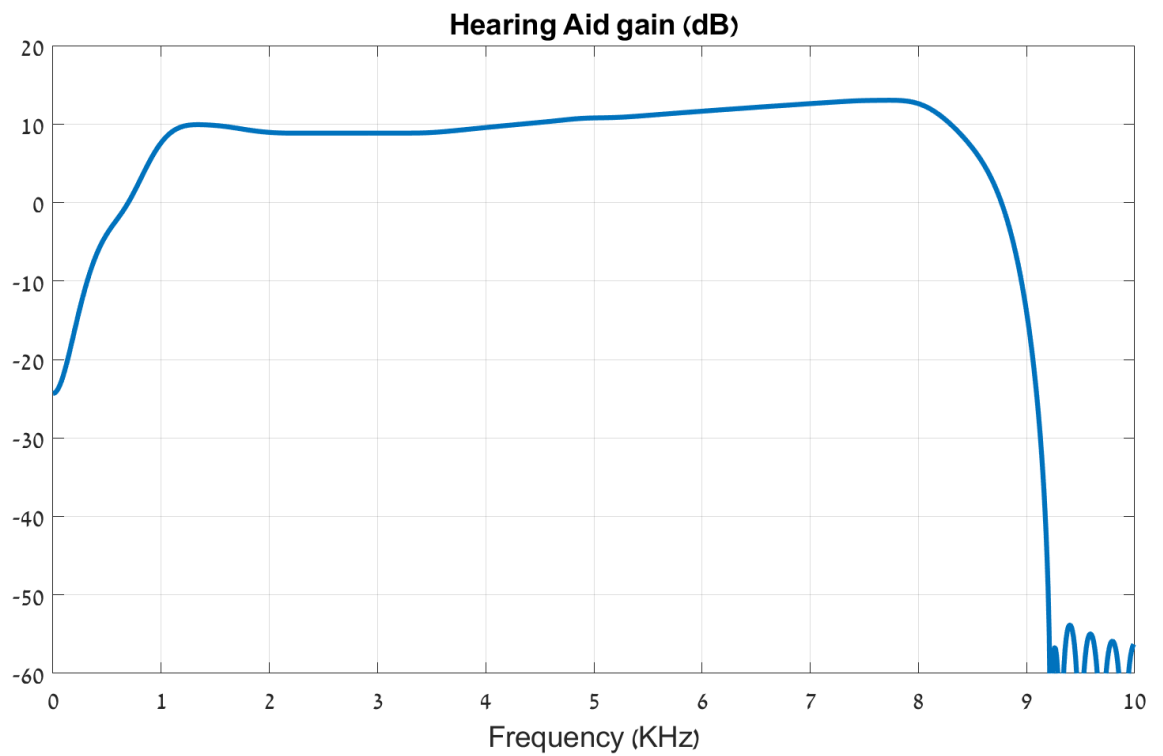
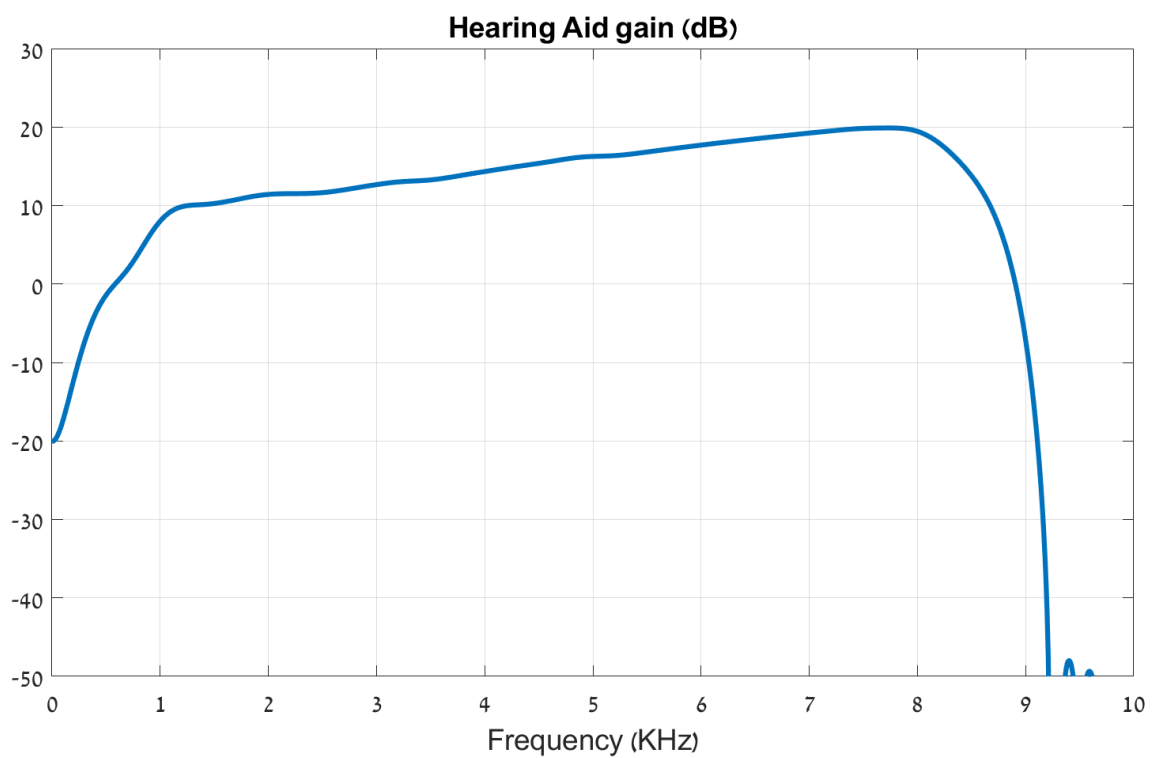


Figure 7.4: show different Audiogram HL for patient with OHC damage, without aid and with different aids prescriptions, described in 7.4 calculated by the program

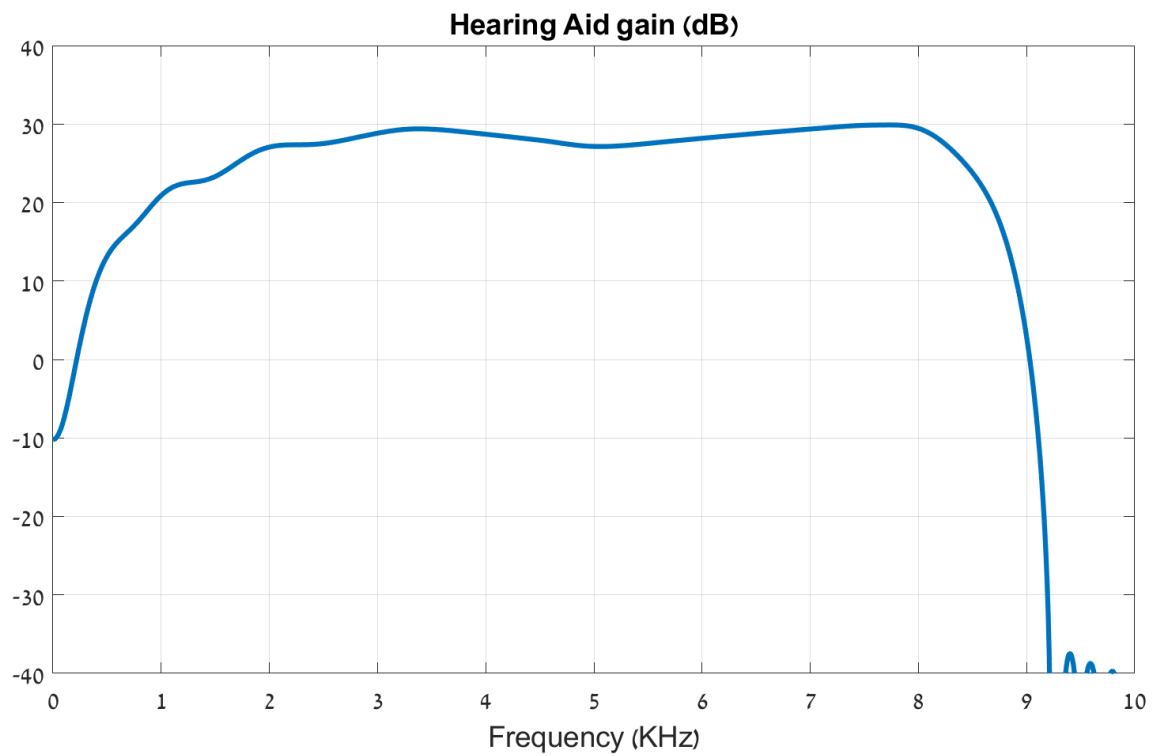


(a) NAL Revised

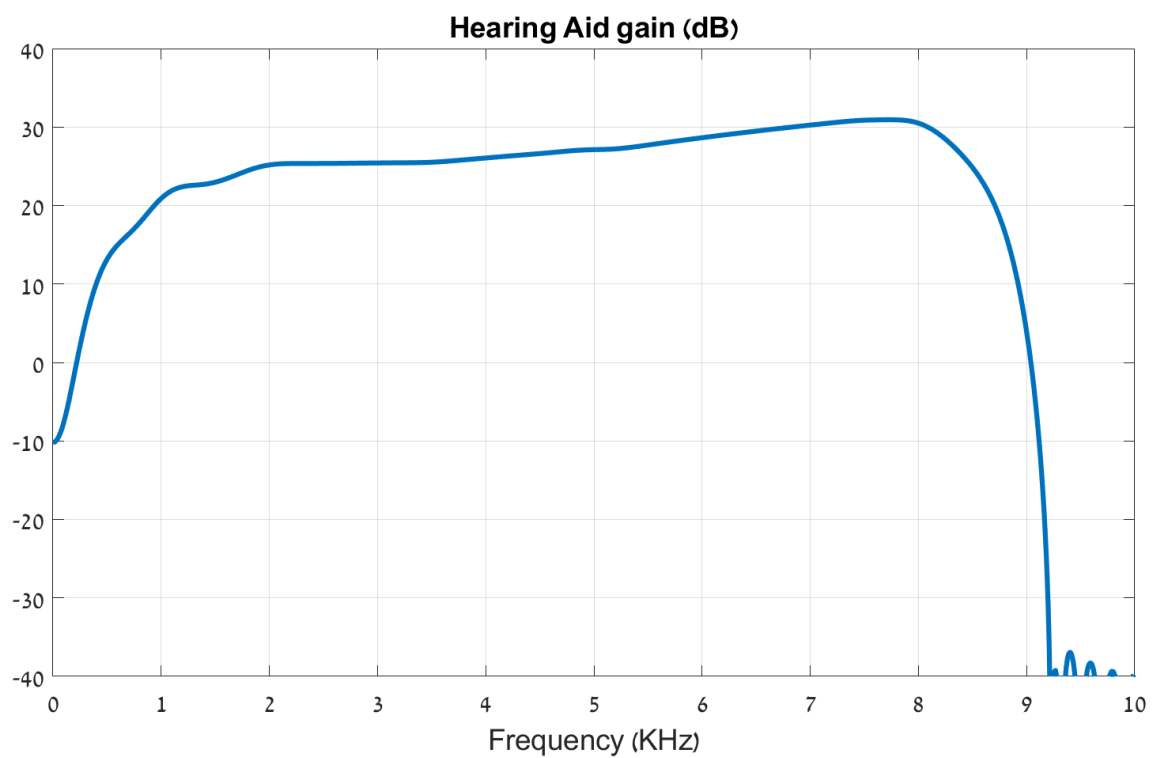


(b) POGO

Figure 7.5: Compare the 4 hearing aids prescriptions with Fig. 7.6, Pogo is the only one that accentuate the higher frequencies

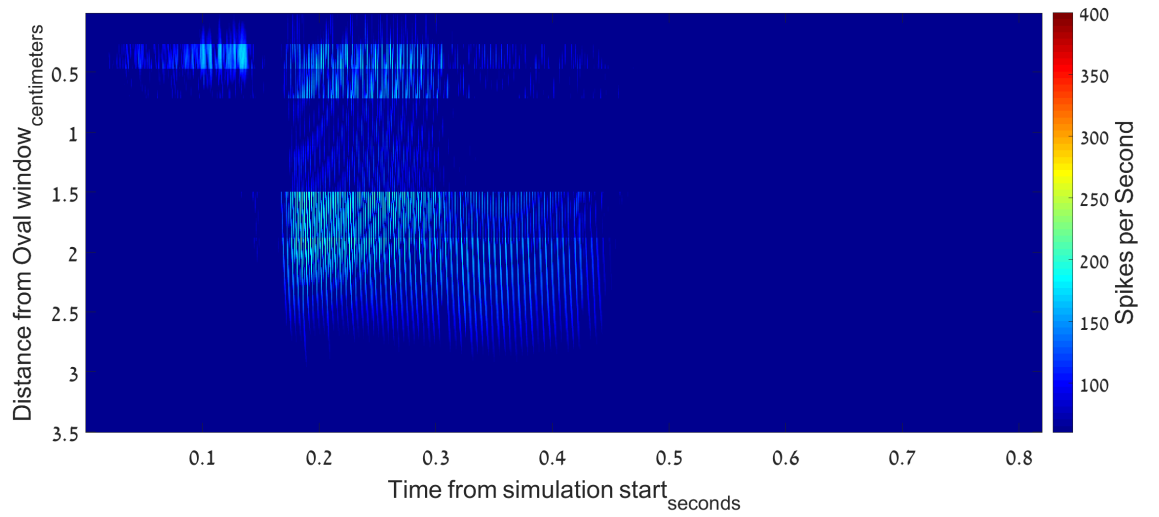


(a) Berfer Behind The Ear

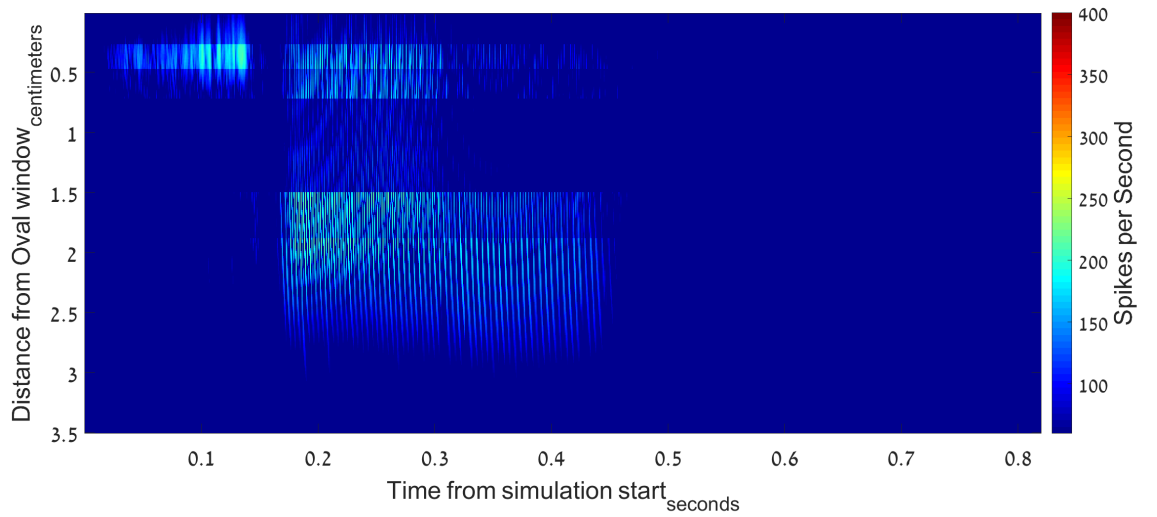


(b) Berger Inside The Ear

Figure 7.6: Compare the 4 hearing aids prescriptions with Fig. 7.5, Berger has much more amplification.

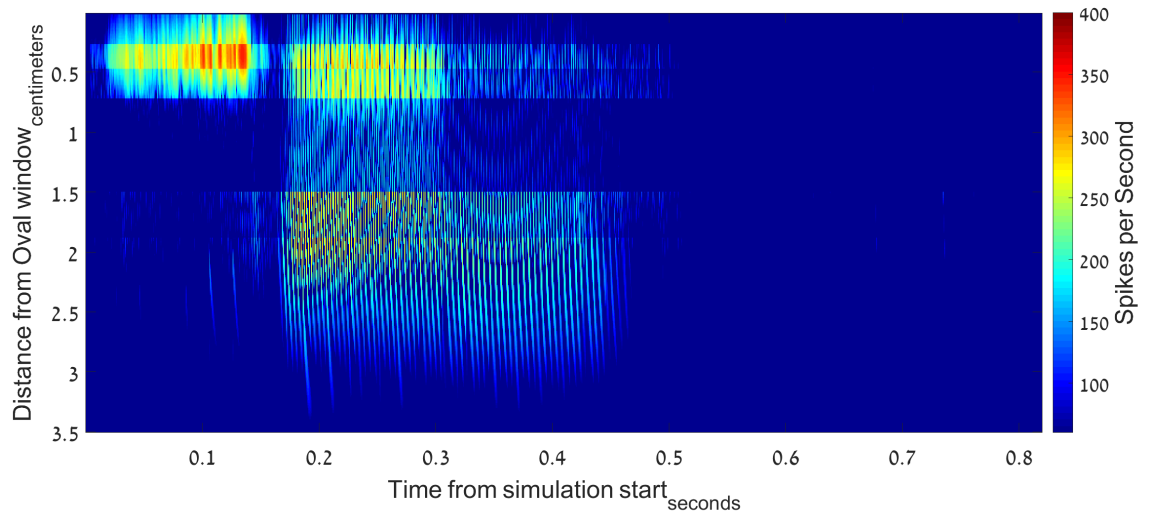


(a) NAL Revised

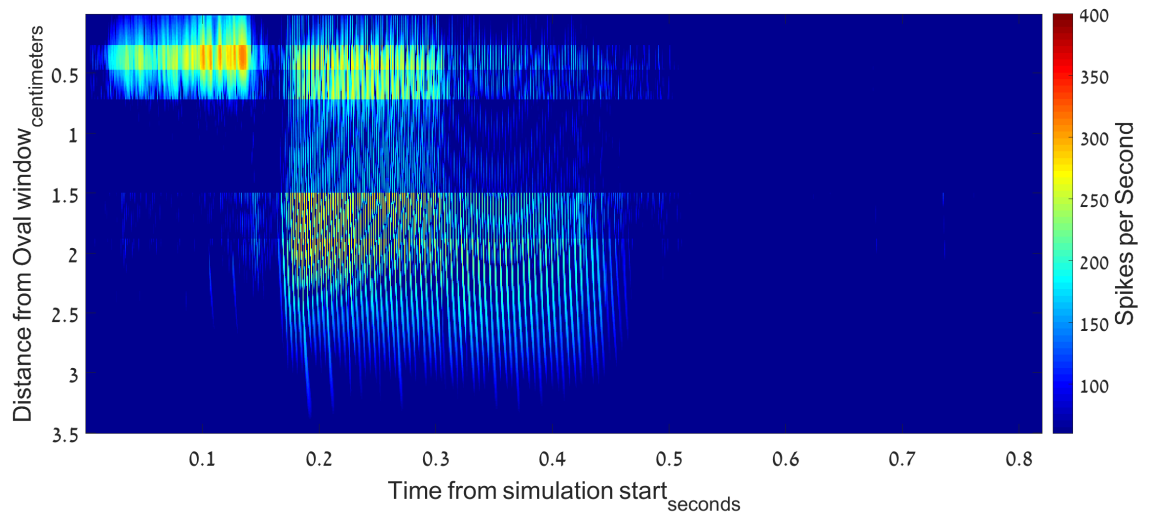


(b) POGO

Figure 7.7: Comparative results of ANR for Hebrew word SHEN, spoken by female at comfortable hearing level, with different aids, response without aid shown at Fig. 7.2(b). both POGO and NAL methods are under severely amplified, the SH consonant response is effectively disappeared, Berger methods shown at Fig. 7.8



(a) Berfer Behind The Ear

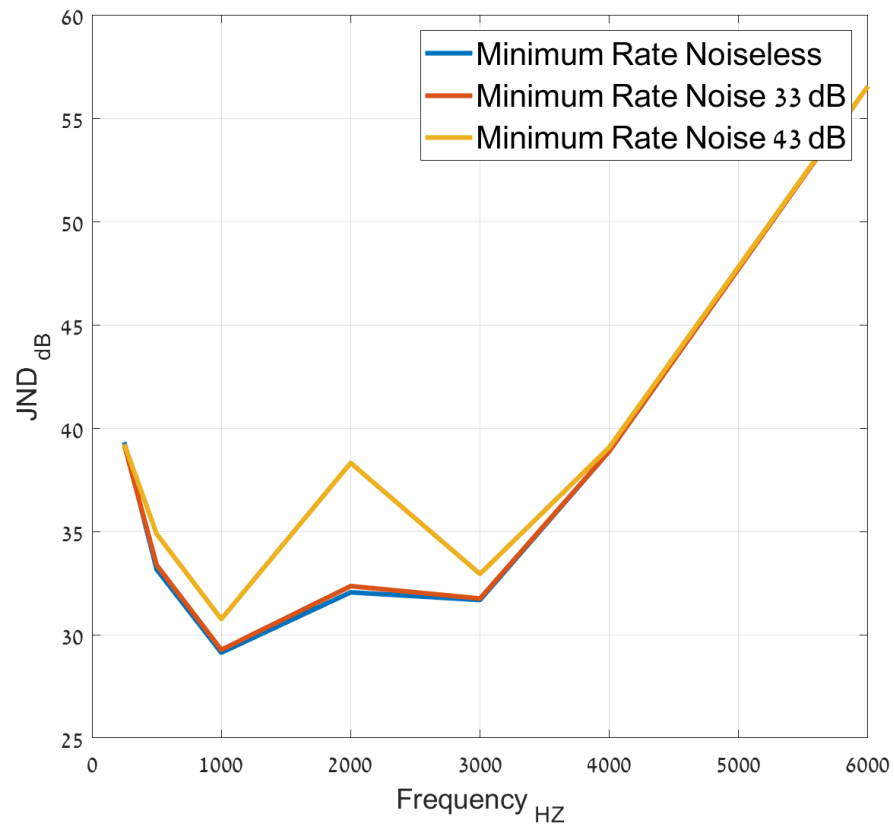


(b) Berger Inside The Ear

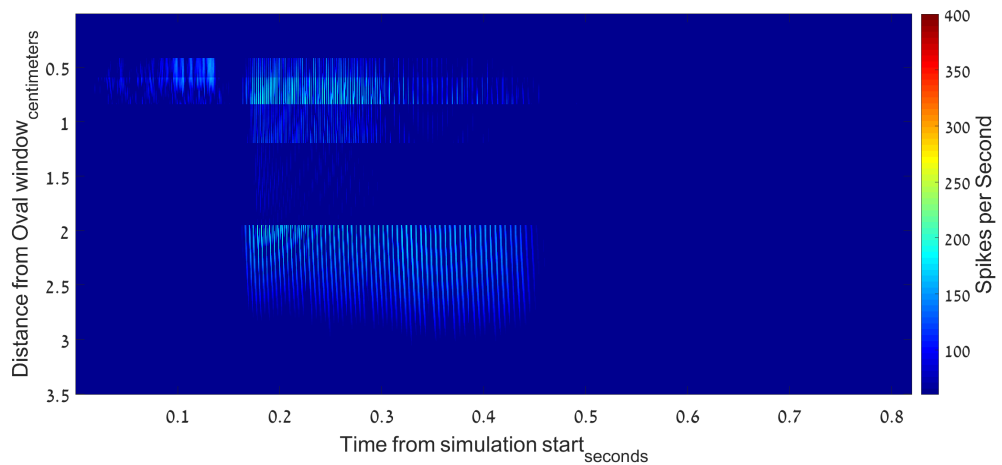
Figure 7.8: Comparative results of ANR for Hebrew word SHEN, spoken by female at comfortable hearing level, with different aids, response without aid shown at Figure 7.2(b). While the Berger methods give better results than NAL and POGO shown at Fig. 7.7, it still significantly different from healthy cochlea. this makes OHC damage unfit for hearing aids to fix.

7.5 Hearing Aid, Combined OHC and IHC damage

as oppose to the patient evaluated at 7.4 who has mostly OHC damage but almost none IHC, we will evaluate effectiveness of different prescriptions on patient with moderate damage for both IHC and OHC.



(a) OHC at 40% and IHC at 68% JND



(b) OHC at 40% and IHC at 68% ANR

Figure 7.9: Model with OHC 40% and IHC 68% ANR when compared to Normal Hearing Patient shown at Fig. 7.1. Response for the Hebrew word SHEN spoken by female at comfortable sound level, compare it to the response of patient with sever OHC damage to the same signal at Fig. 7.2(b), JND degrading is more significant for the lower frequency, as seen both for the SPL audiogram at Fig. 7.9(a) and for the ANR at Fig. 7.9(b), where for this patient very low response for the SH consonant remain, the low frequency component of the E vowel is much weaker that other patient.

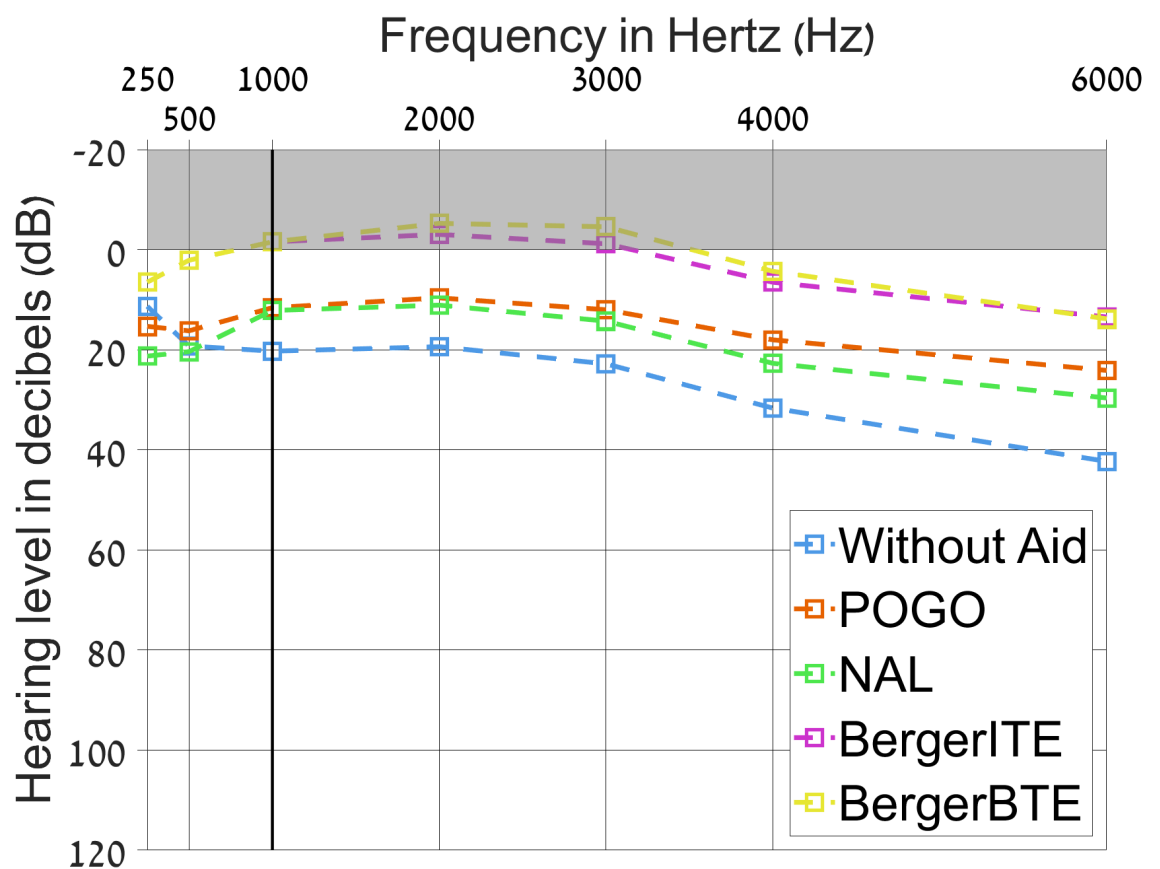
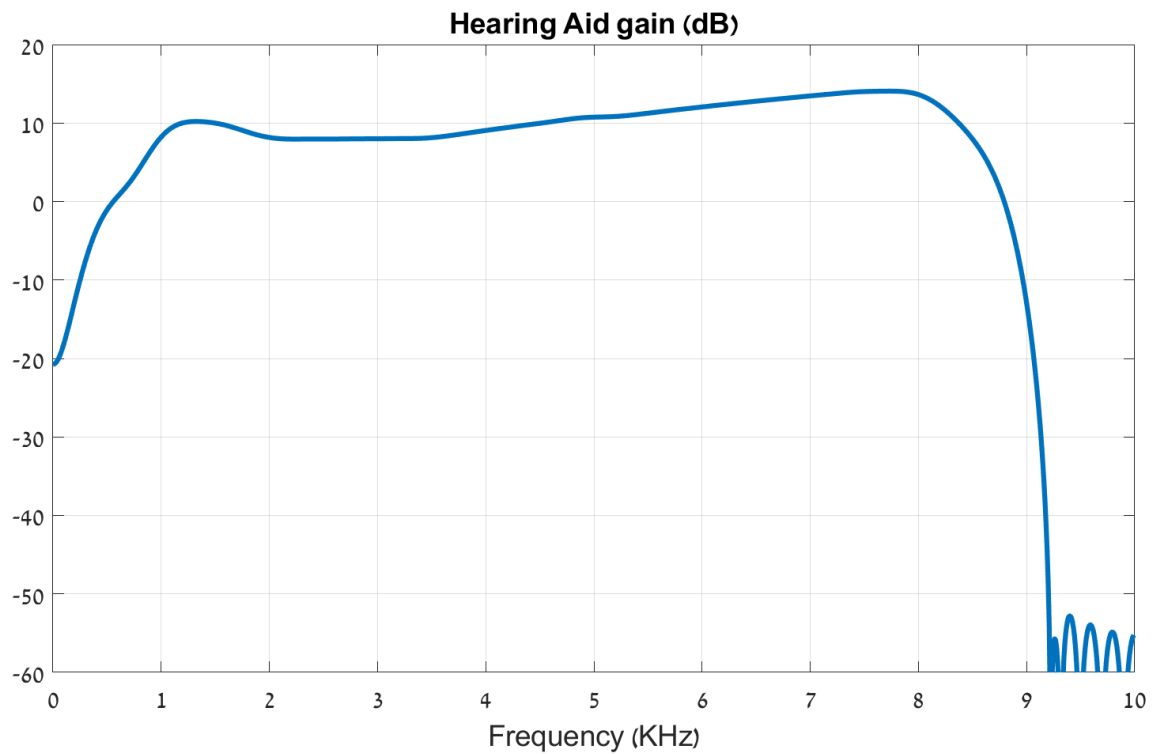
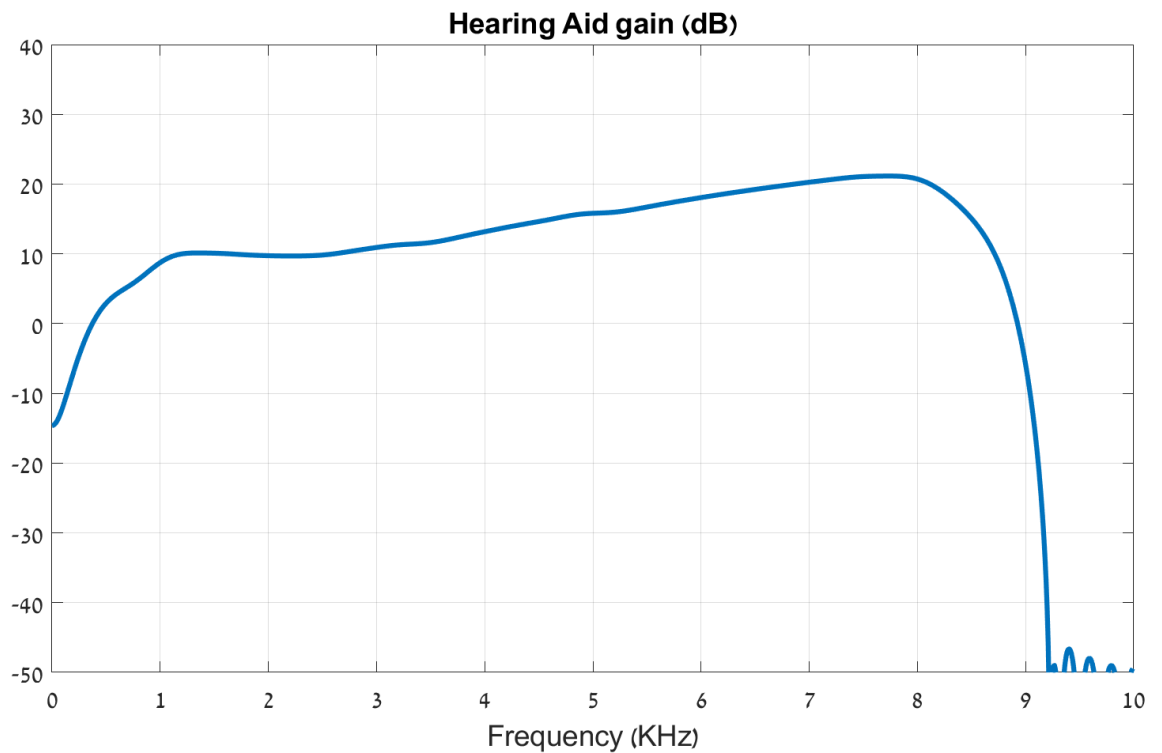


Figure 7.10: The Comparison for patient Audiogram HL with different prescriptions, major differences from Figure 7.10 are that the critical frequencies 2-4KHz are amplified much more than lower. this both prevent low pass noise amplification and avoid upward masking from the signal lower bands components.

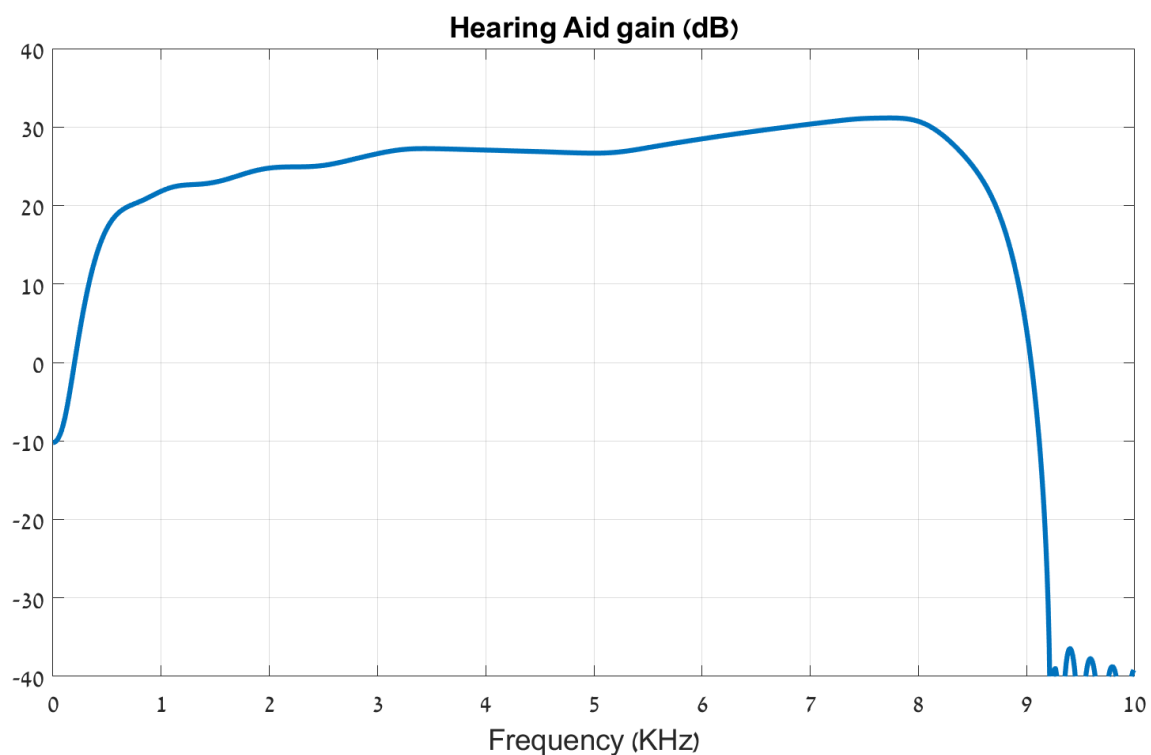


(a) NAL Revised

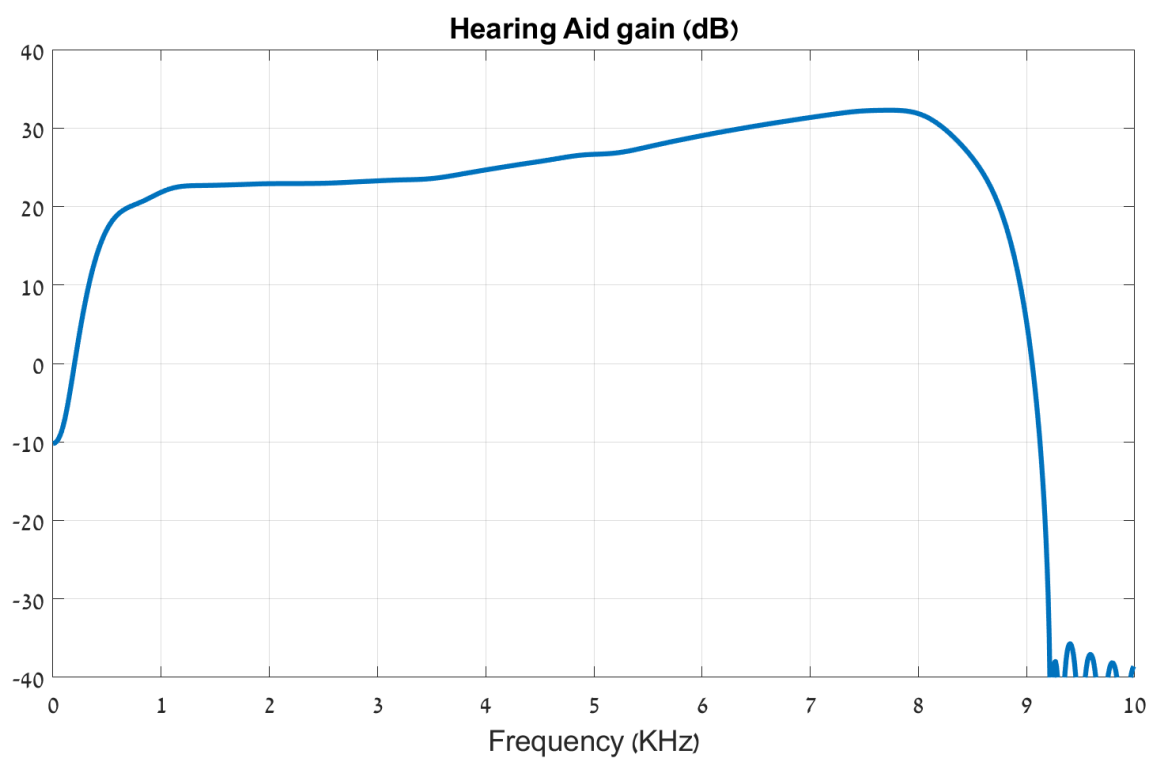


(b) POGO

Figure 7.11: Compare the 4 hearing aids prescriptions, Berger methods shown at Fig. 7.12. NAL and POGO significantly under amplify at higher frequencies relative to berger methods

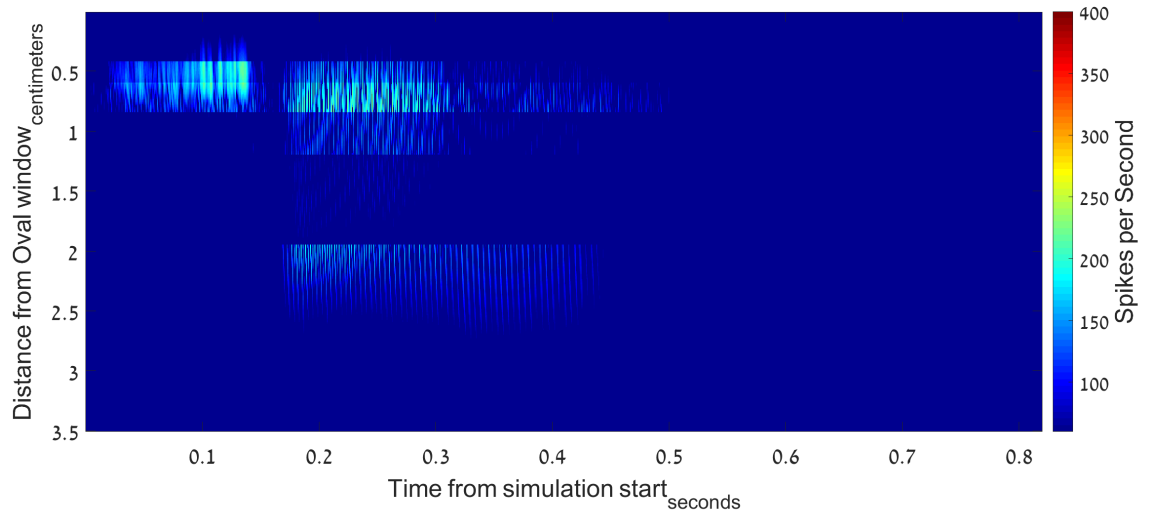


(a) Berfer Behind The Ear

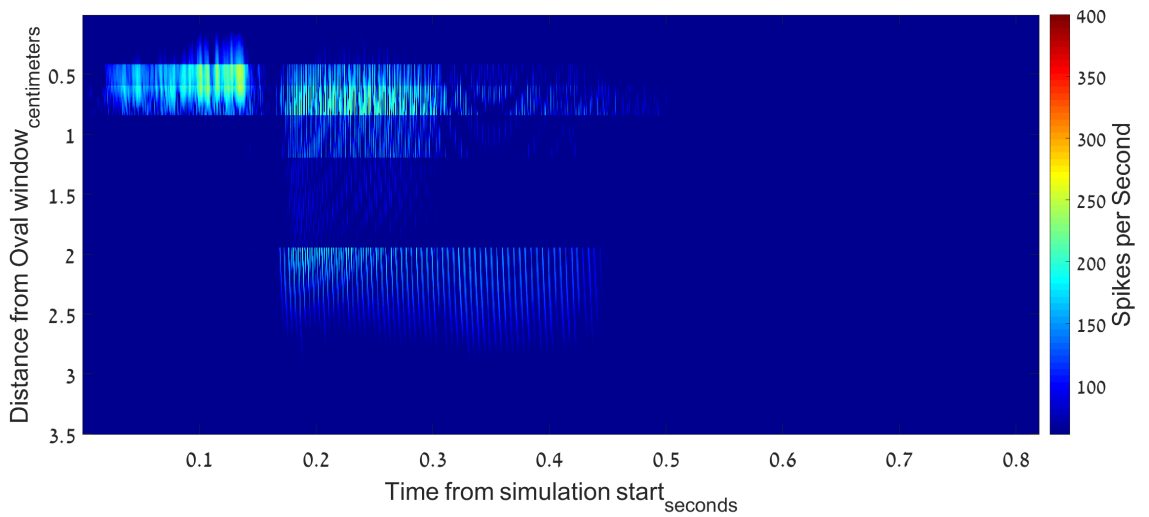


(b) Berger Inside The Ear

Figure 7.12: Compare the 4 hearing aids prescriptions, NAL and POGO shown at Fig. 7.11, Berger has much more amplification and pogo is the only one that accentuate the higher frequencies

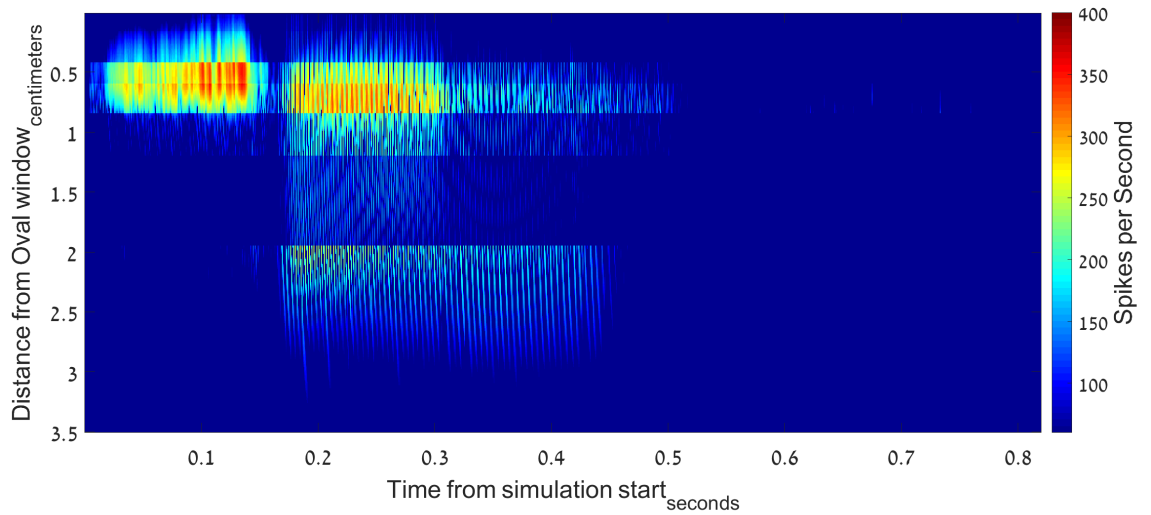


(a) NAL Revised

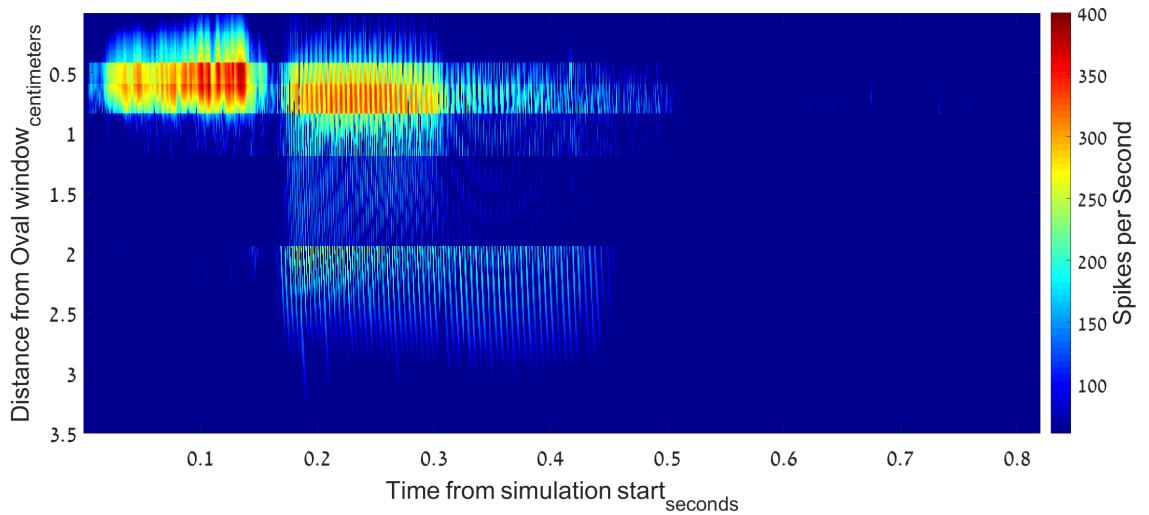


(b) POGO

Figure 7.13: Comparative results of ANR for Hebrew word SHEN, spoken by female at comfortable hearing level, with different aids, response without aid shown at Figure 7.9(b). The NAL and POGO methods amplification is insufficient



(a) Berger Inside The Ear



(b) Berfer Behind The Ear

Figure 7.14: Comparative results of ANR for Hebrew word SHEN for Berger Behind and Inside the Ear, spoken by female at comfortable hearing level, with different aids, response without aid shown at Figure 7.9(b). This methods gives much better amplification from POGO and NAL shown at Fig. 7.13

Chapter 8

Run time Comparison of fully Parallel Algorithm against Only BM velocity Parallelized and full Serial Algorithm

8.1 Run Times for different configurations

Table 8.1 show comparison between several modes of operations and run time for 3 system configuration

1. Oded and Furst 36 work, simulate the cochlear model on CPU without the using GPU, model's demands for $1.36 * 10^{12}$ operations per each second of input signal [39] makes this system unfeasible for real time diagnostics tool for audiologist. some of the time measurements extrapolated from smaller runs due to inability wait years for completion. this system however is useful tool to verify our program correctness for chosen inputs.
2. Sabo et al. 40 work improves significantly calculation speed of Basilar Membrane Velocity, the most resource intense part of the algorithm. by copying the results to CPU memory, save it on Hard Drive and load it with Matlab we can than calculate the ANR response. while this approach is superior to the first but it has weakness. copying data from GPU to CPU and than HD takes time, and neces-

sity of reloading it from HD take time too, computing ANR on CPU is also much slower than GPU. our work show at Chapter 4 that due to error in calculating of signal input time with desired precision, the output is not useful to calculate ANR and therefore JND after first 80ms, lower parallelism significantly.

3. our work improved speed of generating database for diagnostics by observing that input for each damage profile is small, few KB at most for OHC/IHC damage across the cochlea and signal power and frequencies for testing and output is also few KB, the audiogram at various levels of input. by implementing the process of calculating the BM velocity + ANR + JND on GPU we avoid the expensive operations of copy large amounts of input data BM velocity to disk and parallelizing ANR,JND. this allowed by setting amount of allocated memory on GPU at run-time and calculating time so precision is constant.

8.1.1 Comparing run-time for different tasks

To examine the run time of our program, we compare it to 2 similar programs that preceded it at several typical tasks. the entire CPU algorithm developed by [36] and for the CPU/GPU combination we took Sabo et al. 39 work and added ANR and JND calculation on the results in Matlab. timing for each task can be shown at Table 8.1. the acceleration factors differences have several reasons with most significant improvement at the generating hearing impairments databases over [40] program. Several of run-times for pure CPU approximated due to very long time required to run them (days for \mathcal{M} , years \mathcal{C}). we also tested that this tasks have linear relationship between their run-times and number of profiles results from serial nature of the CPU.

1. \mathcal{A} , calculating Audiogram for healthy by using [36] method of testing single input power (adjusted from 7.5dB to 50 due to lower SPL Reference) for 4 levels of noise.
2. \mathcal{B} similar to \mathcal{A} but with 10 levels of signal power to implement [43] minimum required power method, note that are our program execution time multiplied by less than 5 due to under utilizing of the GPU at \mathcal{A} since profile signals were too short to fully utilize the GPU.
3. \mathcal{C} to create database we need profiles of damage at small intervals, 1% for OHC and IHC gives 10000 profiles to test.
4. \mathcal{D} examine the effect of single prescription for the Hebrew word "bor", length 0.8 seconds. for pure CPU method, processing time is similar to that of full damage profile. the pure CPU approach preserve relation of 1000 seconds for every input signal due to the processing time dominating every task. for other approaches and short tasks copying data to CPU for graph presentation on Matlab requires more time.
5. \mathcal{E} simulate ANR of multiple words spoken by both male and female for single prescription method, signals total length is 10.3 seconds, note that our method loose lots of its relative improvement due to Matlab rendering time.
6. \mathcal{L} ANR and JND calculation time for \mathcal{B} , since both done on the CPU, pure CPU method can work with 40ms input, but reliance on erroneous GPU calculation demands 4 times more data. input length for tasks \mathcal{B} and \mathcal{L} is 45.44 seconds for

Method task	Pure CPU	BM Velocity on GPU, ANR and JND on CPU	Full GPU process
\mathcal{A}	1150	93	0.31
\mathcal{B}	11500	928	1.35
\mathcal{C}	$1.15 \cdot 10^8$	$9.28 \cdot 10^6$	$1.35 \cdot 10^4$
\mathcal{D}	811	13	0.66
\mathcal{E}	10180	123	20.9
\mathcal{L}	35	135	0.11
\mathcal{M}	$6 * 10^6$	60000	892

Table 8.1: Comparing run-time for different tasks, with CPU only, GPU + CPU and all heavy calculation tasks on GPU measured in seconds.

combined method and 11.4 for other methods.

7. \mathcal{M} test JND improvement as function of signal time detailed at 8.3. note that we approximate run time on pure CPU as multiplication of $\frac{\#profiles}{4}$ needed by the amount of run-time for 4 profiles tested at \mathcal{A} .

We have tested the pure CPU version of the algorithm on Single core described in Table C.1 For the GPU implementation we ran the simulations on NVidia GTX1080Ti Table C.2 shows the cards' configuration.

1. fixing the linear interpolation between 2 input samples to ensure time invariance, described at Section 4.1, previous program could process 0.3 *seconds* at every execution but only first 80ms could be used by [43], since each block interval handles 20ms of inputs, the card could only compute 4 block intervals at single kernel run. we shown that optimal occupancy division of 112 intervals (28 SMs, each execute 4 blocks), utilization of the GPU improved by factor of 28 for all tasks.
2. to compensate for previous program calculation error, JND calculations required

averaging 160ms of input to approach. we have shown however that using less than 40ms and multiplication factor at Section 6.4. allow us improve factor of 4, but only for database generation.

3. both previous works used to calculate ANR on the CPU this required copy of BM velocity to memory and than hard drive + slow calculation on CPU, by uploading input of frequencies + powers + noise power + damage profile (which is few KB) to the GPU and at the end of JND calculation downloading from GPU memory

8.2 Acceleration from work-flow modifications

We examined in Section 6.7 changes to the work-flow that allow us to avoid both memory allocations and releases (except limited number of times). We can see the run time saved for each run in Fig. 8.1, larger buffers make this difference more significant. We also replaced Reading/Writing to HD by RAM trough Matlab Application Programming Interface, API, time saved shown in Fig. 8.2 and is also relative to buffer size.

8.3 Additional Experiment

This improvement of speed allow for much more experiments in short time than the old programs. We show example of Fig. 8.3.

we have examined the model assumption that calculation of JND by Cramer Raw lower Bound with RMS method [14] will reflect approximately JND for every length of signal.

however JND improves as function of time much beyond NH person. we added the

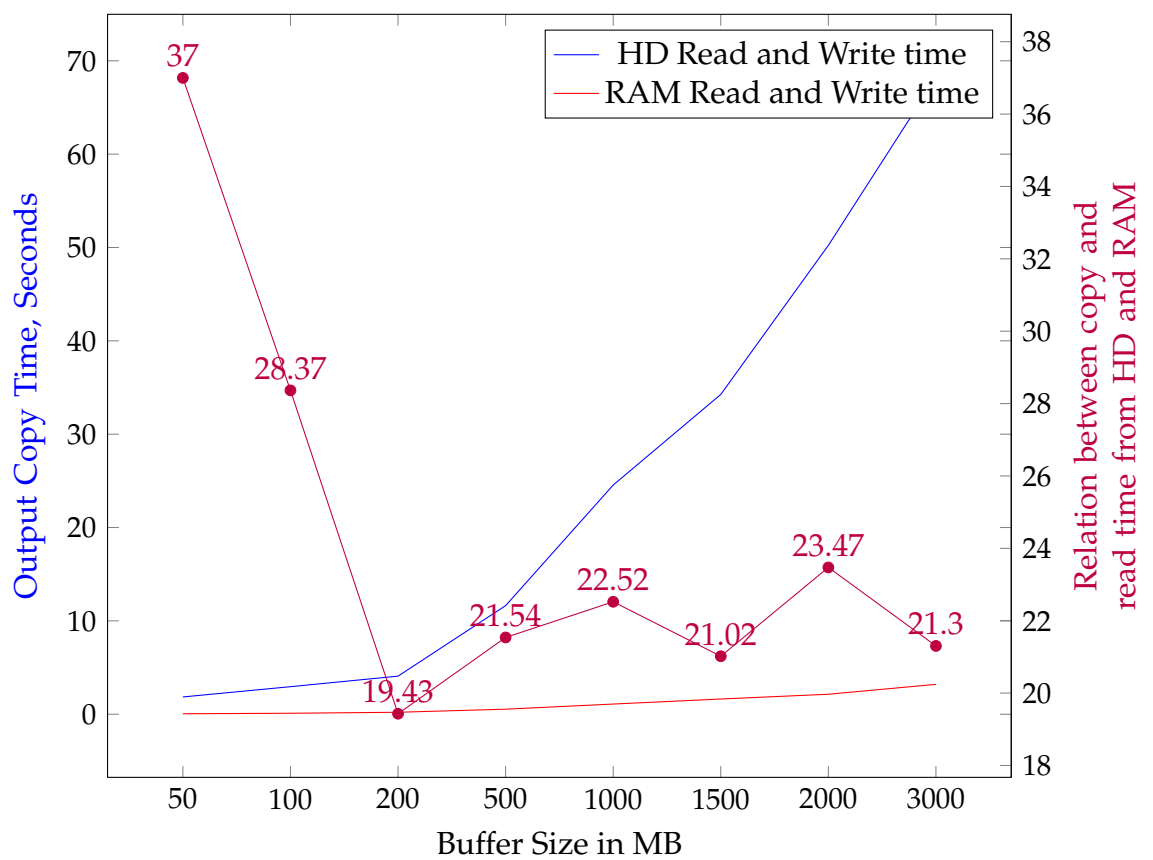


Figure 8.1: Comparison for output copy time, HD and RAM for different buffer sizes, The left Y has the combined time of write be the program and read by Matlab the results. the right Y axis has relation between those times for HD and RAM

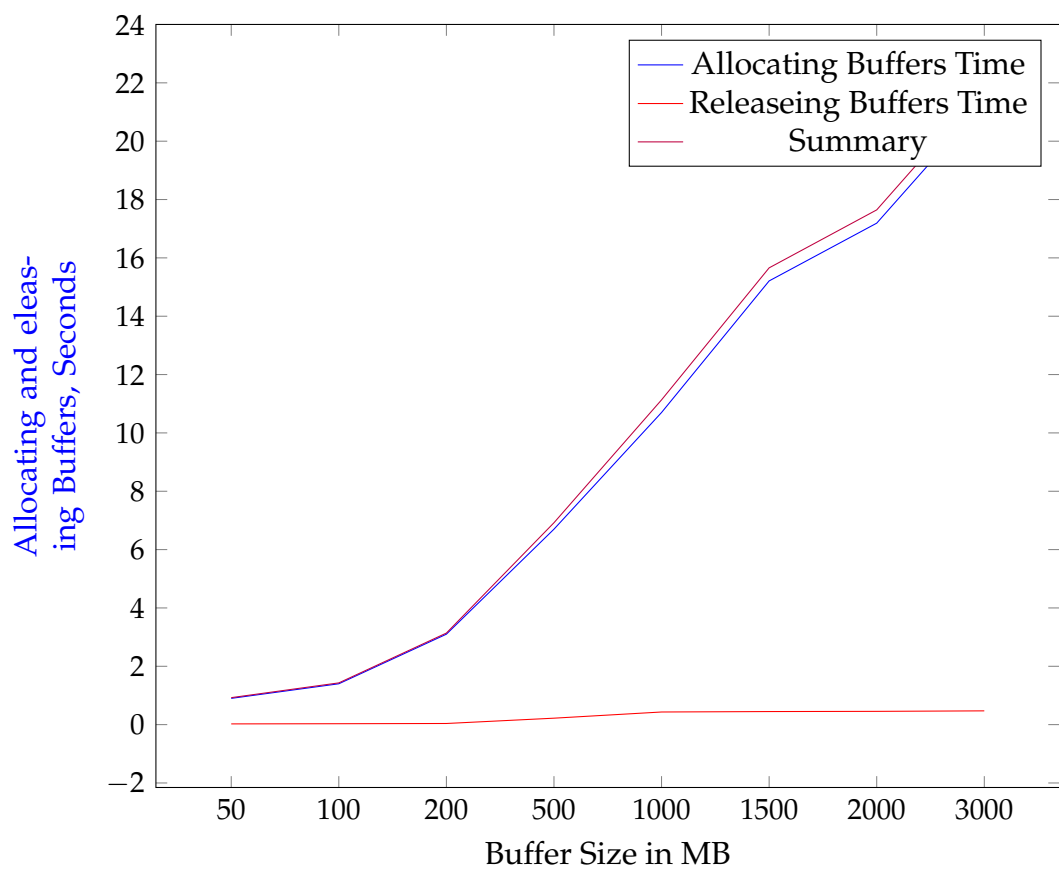


Figure 8.2: Comparison for output copy time, HD and RAM for different buffer sizes, The left Y has the combined time of write be the program and read by Matlab the results. the right Y axis has relation between those times for HD and RAM

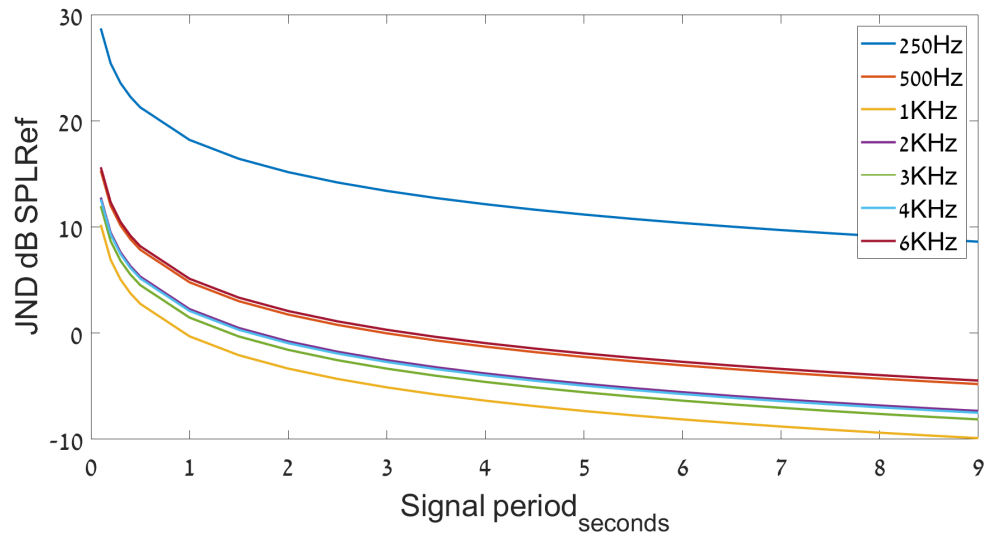


Figure 8.3: test JND for healthy patient as function of signal time. JND for every tested tone measured as function of interval. 22 intervals were 100 to 500msec by jump of 100msec. and 1 to 9 seconds by jump of half second. 10 levels of power 10 to 100dB for 7 frequencies. accumulates to 6055 seconds of audio. times shown at 8.1, \mathcal{M} .

factor k_{JND} described at Section 6.6 to compensate.

Chapter 9

Conclusions

The purpose of this work was to design an efficient tool for audiologist by using parallel algorithm. we took [39] work and expand it to measure ANR and JND. to gain real time response, we implement both the ANR and JND on GPU which improve speed by 200%. but our main improvement was to change input calculation such that precision will not decrease when time increase due to limitations of the 32bit floating point standard. this allows us using the GPU to calculate 2.24 seconds of audio instead mere 80msec, 28 factor improvement from parallelism. the original program permit single signal evaluation by adding overlap for the interval, by implementing selection mechanism to prevent some of the blocks from calculating the overlap, allow us to solve for multiple signals at once. additional observation that [14, Eq. 17] allow emulate JND of single tone at 0.15 seconds by test interval of 40msec and multiply by factor allow further optimization for auditory damage to JND database construction speed. we also allow to generate same signal wave form and noise wave form at different powers on the GPU, preventing creating long signal on the CPU. we also modified program to share memory with Matlab, an improvement by factor of 10 to the run time. this improvements aggregate to multiply 720 of database generation, therefore [43] can cre-

ate 1000 profiles under 4 hours instead of ~3.5 months. the algorithm os searching the database + real-time measurement of effectiveness for different prescription methods permit usage for clinical audiological diagnosis and hearing aid fitting, this tool can be used for fast adjusting of hearing aids and reduce return rate due to unfit prescription. improvement of hearing aid fitness should also help CHH that need proper hearing to develop communications skills.

Appendices

Appendix A

Cochlear Constants

Parameter	Value	Description
K_{bm}	$1.282 \cdot 10^4 e^{-1.5x}$	Basilar membrane stiffness per unit area [gr/cm^2s^2]
R_{bm}	$0.25 e^{-0.6x}$	Basilar membrane resistance per unit area [gr/cm^2s]
M_{bm}	$1.286 \cdot 10^{-6} e^{1.5x}$	Basilar membrane mass per unit area [gr/cm^2]
K_{tm}	$3.97 \cdot 10^5 e^{-3.06x}$	Tectorial membrane stiffness per unit area [gr/cm^2s^2]
R_{tm}	$0.25 e^{-0.6x}$	Tectorial membrane resistance per unit area [gr/cm^2s]
α	$1 \cdot 10^{-6}$	Peak to peak electromotility displacement [cm]
ω_{ohc}	1000	Outer hair cells cutoff frequency [Hz]
ω_{ow}	1500	Oval window cutoff frequency [Hz]
σ_{ow}	0.5	Oval window aerial density [gr/cm^2]
C_{ow}	$6 \cdot 10^{-3}$	Coupling of oval window to perilymph [<i>none</i>]
Γ_{ME}	21.4	Mechanical gain of ossicles [<i>none</i>]
γ_{ow}	$20 \cdot 10^3$	Oval window resistance [$1/s$]
ρ	1	Perylimph density [gr/cm^3]
β	0.003	Width of the basilar membrane [cm]
A	0.5	Cross—sectional area of the cochlea scalae [cm^2]
L_{co}	3.5	Cochlear length [cm]
$Frequency_{pass}$	600	h_{ihc} low pass filter transfer function pass band frequency (HZ)
$Frequency_{stop}$	1600	h_{ihc} low pass filter transfer function stop band frequency (HZ)
$Attenuation_{pass}$	3dB	h_{ihc} low pass filter $Frequency_{pass}$ Gain (dB)
$Attenuation_{pass}$	30dB	h_{ihc} low pass filter $Frequency_{stop}$ Gain (dB)
$\lambda_{spont}^{(HSR)}$	60	rate of spikes per second for HSR AN

$\lambda_{spont}^{(MSR)}$	3	rate of spikes per second for MSR AN
$\lambda_{spont}^{(LSR)}$	0.1	rate of spikes per second for LSR AN
ω_H	0.61	weight of HSR AN
ω_M	0.23	weight of MSR AN
ω_L	0.16	weight of LSR AN
SPL_{ref}	$2 \cdot 10^{-5}$	sound pressure level (Pascal) physical from [49]
SPL_{ref}	$1.5 \cdot 10^{-8}$	sound pressure level (Pascal) set by [43]
η_{AC}	1	(V/s/cm) multiplication factor for AC component of IHC voltage
η_{DC}	100	(V/cm) multiplication factor for DC component of IHC voltage

Table A.1: Parameters for cochlear equations solution.

Appendix B

Architecture Features Comparison

SM Generation	Fermi		Kepler				Maxwell			Pascal		
SM Architecture	2.0	2.1	3.0	3.2	3.5	3.7	5.0	5.2	5.3	6.0	6.1	6.2
Maximum number of resident grids per device (concurrent kernel execution)	16			4	32				16	128	32	16
Maximum dimensionality of grid of thread blocks	3											
Maximum x-dimension of a grid of thread blocks	65535		$2^{31} - 1$									
Maximum y-, or z-dimension of a grid of thread blocks	65535											
Maximum dimensionality of thread block	3											
Maximum x- or y-dimension of a block	1024											
Maximum z-dimension of a block	64											
Maximum number of threads per block	1024											
Warp Size	32											

Maximum number of resident blocks per multiprocessor	8	16			32			
Maximum number of resident warps per multiprocessor	48	64						
Maximum number of resident threads per multiprocessor	1536	2048						
Number of 32-bit registers per multiprocessor	32K	64K		128K	64K			
Maximum number of 32-bit registers per thread block	32K	64K	32K	64K		32K	64K	32K
Maximum registers per thread	63		255					
Maximum amount of shared memory per multiprocessor	48K			112K	64K	96K	64K	96K
Maximum amount of shared memory per thread block	48K							
Number of shared memory banks	32							
Amount of local memory per thread	512K							
Constant Memory size	64K							
Cache working set per multiprocessor for constant memory	8K						4K	8K

Maximum number of instructions per kernel	512 million
---	-------------

Table B.1: NVidia GPUs Technical specifications parameters summarized from [34, Pg. 218-220]

SM Generation	Fermi		Kepler			Maxwell		Pascal	
SM Architecture	2.0	2.1	3.0	3.5	3.7	5.0	5.2	6.0	6.1,6.2
Number of ALU lanes for integer and single-precision floating-point arithmetic operations	32	48	192			128		64	128
Number of special function units for single-precision floating-point transcendental functions	4	8	32					16	32
Number of warp schedulers	2		4				2		4
Max number of instructions issued at once by a single scheduler	1	2				1			

Table B.2: NVidia GPUs Architecture specifications parameters summarized from [34, Pg. 83-84]

Appendix C

Hardware Used

This Appendix contains Hardware used for tests. for the GPU all experiments used the GTX1080TI unless written otherwise.

CPU	Intel i7-4770 4 cores,8 threads
CPU Clock	3.9 GHz
Cache	8 MByte L3
RAM Size	16 GByte (DDR2)
RAM Clock	1600 MHz
HD	Seagate Barracuda 7200.14 3 TB

Table C.1: Processor configuration for sequential, single-core computation.

Card	NVIDIA GeForce 1080TI	NVIDIA GeForce 760
GPU Family	GP102	GK104
Architecture	Pascal (SM 6.1)	Kepler (SM 3.0)
Streaming Multi Processors (SM)	28	6
GPU cores per SM	128	192
GPU cores Summary	3584	1152
GPU clock	1683 MHz	1150 MHz
Memory clock	5505 MHz	3004 MHz
RAM size	11 GByte	2 GByte
Frame Buffer Bandwidth	484 GB/sec	192 GB/sec

Table C.2: Hardware configuration for the parallel code.

Bibliography

- [1] How does the ear work? URL <https://www.news-medical.net/health/How-Does-the-Ear-Work.aspx>.
- [2] Nvidia nsight visual studio edition 5.4 user guide. URL https://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_1080_Whitepaper_FINAL.pdf.
- [3] Iso226:2003, 2003. URL http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34222.
- [4] M. J. Borsboom and M. A. Viergever. The stretching nonlinearity of the basilar membrane in a cochlear model. *Hearing Research*, 2(34):485 – 492, 1980. ISSN 0378-5955. doi: 10.1016/0378-5955(80)90085-4. URL <http://www.sciencedirect.com/science/article/pii/0378595580900854>.
- [5] W. E. Brownell, C. R. Bader, D. Bertrand, and Y. de Ribaupierre. Evoked mechanical responses of isolated cochlear outer hair cells. *Science (New York, N.Y.)*, 227(4683):194–6, Jan. 1985. ISSN 0036-8075. URL <http://www.ncbi.nlm.nih.gov/pubmed/3966153>.
- [6] A. Cohen and M. Furst. Integration of outer hair cell activity in a one-dimensional

- cochlear model. *The Journal of the Acoustical Society of America*, 115(5):2185–2192, 2004.
- [7] R. Courant, K. Friedrichs, and H. Lewy. ber die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928. ISSN 0025-5831. doi: 10.1007/BF01448839. URL <http://dx.doi.org/10.1007/BF01448839>.
- [8] P. Dallos. Organ of corti kinematics. *J. Assoc. Res. Otolaryngol*, 4:416–421, 2003.
- [9] S. P. Davis A, S. D. Ferguson M, and G. I. Acceptability, benefit and costs of early screening for hearing disability: a study of potential screening tests and models. *Health Technol Assess*, October 2007. URL <https://www.ncbi.nlm.nih.gov/pubmed/17927921/>.
- [10] H. Dillon. *Hearing Aids Introductory Concepts*. 2001.
- [11] H. Dillon. *Sandlin’s Textbook of Hearing Aid Amplification: Technical and Clinical Considerations*. 2014.
- [12] M. . Fortnum. Why do people fitted with hearing aids not wear them? *International Journal of Audiology*, May 2013. doi: 10.3109/14992027.2013.769066. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3665209/>.
- [13] M. . Fortnum. Why do people fitted with hearing aids not wear them, table 2. *International Journal of Audiology*, May 2013. doi: 10.3109/14992027.2013.769066. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3665209/table/T2/>.

- [14] M. Furst. Cochlear model for hearing loss. *Procedia Computer Science*, 18 (0):682 – 691, 2015. ISSN 1877-0509. doi: <http://dx.doi.org/10.1016/j.procs.2013.05.232>. URL <http://www.sciencedirect.com/science/article/pii/S187705091300375X>. ;ce:title;2015 International Conference on Computational Science; /ce:title;.
- [15] M. Furst and J. L. Goldstein. A cochlear nonlinear transmission-line model compatible with combination tone psychophysics. *The Journal of the Acoustical Society of America*, 72(3):717–26, Sept. 1982. ISSN 0001-4966. URL <http://www.ncbi.nlm.nih.gov/pubmed/7130530>.
- [16] D. G ddecke and R. Strzodka. Cyclic reduction tridiagonal solvers on GPUs applied to mixed precision multigrid. *IEEE Transactions on Parallel and Distributed Systems*, 22(1):22–32, Jan. 2011.
- [17] P. Gray. Conditional probability analyses of the spike activity of single neurons. *Biophys. J.*, 10:759–777, 1967.
- [18] D. Z. He and P. Dallos. Properties of voltage-dependent somatic stiffness of cochlear outer hair cells. *Journal of the Association for Research in Otolaryngology*, 1(1):64–81, July 2000. ISSN 1525-3961. doi: 10.1007/s101620010006. URL <http://www.springerlink.com/index/10.1007/s101620010006>.
- [19] C. H. Heinz, M.G. and L. Carney. Evaluating auditory performance limits: I. one-parameter discrimination using a computational model for the auditory nerve. *Neural Comput*, pages 2273–2316, 2001.

- [20] M. Heinz. Computational modeling of sensorineural hearing loss. *Computational Models of the Auditory System*, pages 177–202, 2010.
- [21] Z. J. Theory of acoustical action of the cochlea. *J. Acoust. Soc. Am.*, 22(6):778–784, 1950. doi: 10.1121/1.1906689. URL <http://ci.nii.ac.jp/naid/30016063593/en/>.
- [22] J. J. O. P. S. E. A. P. E. W. P. J. Bruce Tomblin, PhD and P. Mary Pat Moeller. The influence of hearing aids on the speech and language development of children with hearing loss. *JAMA Otolaryngol Head Neck Surg*, 140(5):403–409, May 2014. doi: 10.1001/jamaoto.2014.267. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4066968/>.
- [23] Z. W. Jerzy witek, Leszek Borzemski. *Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology*. Springer, 2017, September 2017.
- [24] T. M. John Cheng, Max Grossman. *Proffesional CUDA C Programming*. 2014. URL <http://www.hds.bme.hu/~fhegedus/C++/Professional%20CUDA%20C%20Programming.pdf>.
- [25] Mathworks. C Matrix API. <https://www.mathworks.com/help/matlab/cc-mx-matrix-library.html>, 2018. [Online; accessed 2018].
- [26] Mathworks. dos. <https://www.mathworks.com/help/matlab/ref/dos.html>, 2018. [Online; accessed 2018].

- [27] D. Mountain and A. Hubbard. Computational analysis of hair cell and auditory nerve processes. *Auditory Computation, edited by H.L. Hawkins, T.A. McMullen, A.N. Popper, and R.R. Fay. Springer, pages 121–156, 1995.*
- [28] R. Nagle, E. Saff, and A. Snider. *Fundamentals of differential equations*. Benjamin/Cummings, 1989.
- [29] S. T. Neely and L. Y. W. Outer hair cell electromechanical properties in a nonlinear piezoelectric model. *J. Acoust. Soc. Am.*, 126:751–761, 2009.
- [30] NVIDIA. Kepler tm gk110. 2012. URL <http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf>.
- [31] NVIDIA. *CUDA C Programming Guide, version 4.2*. April 2012. URL http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.
- [32] NVIDIA. Nvidia geforce gtx 980. 2014. URL http://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_980_Whitepaper_FINAL.PDF.
- [33] NVIDIA. *CUDA C Programming Guide, version 7.5*. September 2015. URL http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.
- [34] NVIDIA. *CUDA C Programming Guide, version 8.0*. June 2017. URL http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.
- [35] NVIDIA. *CUDA C Programming Guide, version 8.0*. June 2017. URL http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.

- [36] B. Oded and M. Furst. Timedomain onedimensional cochlear model with integrated tectorial membrane and outer hair cells. Master's thesis, Tel Aviv University, 2011.
- [37] v. S. R. Rieke F., Warland D. and B. W. Spikes exploring the neural code. *MIT Press, Cambridge, Mass, USA.*, 1997.
- [38] P. R. Roth T.N., Hanebuth D. Prevalence of age-related hearing loss in europe: A review. *Euro Arch Otorhinolaryngol.*, 268:1101–1107, 2011. doi: 10.1007/s00405-011-1597-8. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3132411/>.
- [39] D. Sabo, S. Weiss, and M. Furst. Parallel processing of speech reconstruction based on the cochlear model. *Procedia Computer Science*, 18(0): 682 – 691, 2013. ISSN 1877-0509. doi: <http://dx.doi.org/10.1016/j.procs.2013.05.232>. URL <http://www.sciencedirect.com/science/article/pii/S187705091300375X>. ;ce:title;2013 International Conference on Computational Science;/ce:title;.
- [40] D. Sabo, S. Weiss, and M. Furst. A parallel algorithm for a physiological non-linear model of the cochlea. Master's thesis, 2013. URL <http://www.sciencedirect.com/science/article/pii/S187705091300375X>. ;ce:title;2013 International Conference on Computational Science;/ce:title;.
- [41] W. M. Siebert. Frequency discrimination in auditory system,place or periodicity mechanism. *IEEE58*, pages 723–730, 1970.

- [42] s. e. someone and another one. Results of experimental tests in tel ha shomer. *Some Journal*, 57(1):7–24, 2016.
- [43] O. Starwitsky and M. Furst. Cochlear model for hearing loss parameters analysis. Master’s thesis, Tel Aviv University, 2017.
- [44] H. B. Stelmachowicz PG1, Pittman AL and L. DE. Aided perception of /s/ and /z/ by hearing-impaired children. *Ear and Hearing*, 23(4), August 2002. URL <https://www.ncbi.nlm.nih.gov/pubmed/11681394>.
- [45] B. R. Stiles DJ and M. KK. The speech intelligibility index and the pure-tone average as predictors of lexical ability in children fit with hearing aids. *Journal of speech, language, and hearing research : JSLHR.*, 55(3), June 2012. doi: 10.1044/1092-4388(2011/10-0264). URL <https://www.ncbi.nlm.nih.gov/pubmed/22223888>.
- [46] L.-P. E. O. L. Sumner, C. and R. Meddis. A revised model of the inner-hair cell and auditory-nerve complex. 2002.
- [47] C. L. Talmadge, a. Tubis, G. R. Long, and P. Piskorski. Modeling otoacoustic emission and hearing threshold fine structures. *The Journal of the Acoustical Society of America*, 104(3 Pt 1):1517–43, Sept. 1998. ISSN 0001-4966. URL <http://www.ncbi.nlm.nih.gov/pubmed/9745736>.
- [48] V. M. S. H. e. a. Teresa Y.C. Ching, Harvey Dillon. Outcomes of early- and late-identified children at 3 years of age: Findings from a prospective population-based study. *Ear and Hearing*, 34(5), September 2013. doi: 10.1097/

- AUD.0b013e3182857718. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3681915/>.
- [49] I. unknown. standards for physical units. 2002.
- [50] R. J. White, J.A. and A. Kay. Channel noise in neurons. trends. *Neurosci.*, 23: 131–137, 2000.
- [51] Wikipedia. Hair cells.
- [52] Wikipedia. Scheduling (computing). [https://en.wikipedia.org/wiki/Scheduling_\(computing\)](https://en.wikipedia.org/wiki/Scheduling_(computing)), 2013. [Online; accessed 2013].
- [53] I. Yavneh. On red black SOR smoothing in multigrid. *SIAM J. Sci. Comput*, 17: 180–192, 1994.
- [54] J. B. F. V. A. K. A. yvind Nordvik, Peder O. Laugen Heggdal and H. J. Aarstad. Generic quality of life in persons with hearing loss: a systematic literature review. *BMC Ear Nose Throat Disord*, 18, January 2018. doi: 10.1186/s12901-018-0051-6. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5778781/>.
- [55] A. G. A. Yvonne S. Sininger, PhD and B. Elizabeth Christensen. Auditory development in early amplified children: Factors influencing auditory-based communication outcomes in children with hearing loss. *Ear and Hearing*, 31(2):166–185, April 2010. doi: 10.1097/AUD.0b013e3181c8e7b6. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2836405/>.

- [56] M. L. O. D. F. B. Zheng, J. and P. Dallos. Prestin, the motor protein of outer hair cells. *Audiol. Neurotol*, 7:9–12, 2002.
- [57] B. I. N. P. Zilany, M.S.A. and L. Carney. A phenomenological model of the synapse between the inner hair cell and auditory nerve: long-term adaptation with power-law. 126:2390–2412, 2009.
- [58] G. Zweig, R. Lipes, and J. Pierce. The cochlear compromise. *The Journal of the Acoustical Society of America*, 59(4):975–982, 1976.