MICHAL HADRAVA

NETWORK OF NONLINEAR OSCILLATORS AS A MODEL OF MUSIC PERCEPTION IN HUMAN LISTENERS

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Michal H a d r a v a

Study programme: Open Informatics

Specialisation: Artificial Intelligence

Title of Diploma Thesis: Network of Nonlinear Oscillators as a Model of Music Perception in Human Listeners

Guidelines:

Familiarize yourself with the subject of Gradient-Frequency Neural Networks (GFNN). Make experiments in an attempt to answer especially the following questions:

1. Which of the tonal relationships perceived by humans while listening to music can be modeled using GFNN? Is it confined to the relationship between Subtonic and Tonic in the major scale or does it encompass, e.g., chromatic scale degrees?

2. When stimulated with modulating musical phrase, does GFNN reflect the change of a key?

The model will be tested against existing empirical data from experiments if convenient, otherwise rules from standard harmony textbooks will be used.

Bibliography/Sources: Will be provided by the supervisor.

Diploma Thesis Supervisor: Ing. Jan Drchal

Valid until: the end of the summer semester of academic year 2012/2013

L.S.

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 10, 2012

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Michal H a d r a v a

Studijní program: Otevřená informatika (magisterský)

Obor: Umělá inteligence

Název tématu: Síť nelineárních oscilátorů jako model vnímání hudby u člověka

Pokyny pro vypracování:

Nastudujte problematiku Gradient-Frequency Nonlinear oscillator Networks (GFNN). Proveďte experimenty, které se pokusí zodpovědět zejména následující otázky:

1. Které z tonálních vztahů, které člověk vnímá při poslechu hudby, je GFNN schopna modelovat? Dokáže např. GFNN kromě vztahu mezi přirozeným citlivým tónem a tónikou postihnout i vztah mezi umělými citlivými tóny a tónikou?
2. Dokáže GFNN modelovat přehodnocení tonálních vztahů při změně tóniny v průběhu skladby?

Vnímání tonálních vztahů při poslechu hudby bude v experimentech reprezentováno existujícími daty z experimentů s lidskými subjekty, případně, pokud taková data nebudou k dispozici, pravidly harmonie kodifikovanými ve všeobecně uznávaných učebnicích harmonie.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Jan Drchal

Platnost zadání: do konce letního semestru 2012/2013

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2012

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Diploma Thesis

NETWORK OF NONLINEAR OSCILLATORS AS A MODEL OF MUSIC PERCEPTION IN HUMAN LISTENERS

BC. MICHAL HADRAVA

Supervisor: Ing. Jan Drchal

Study Program: Open Informatics

Specialization: Artificial Intelligence

January 3, 2013

Michal Hadrava: *Network of Nonlinear Oscillators as a Model of Music Perception in Human Listeners,* Diploma Thesis, © January 3, 2013

Dedicated to Music as an apology for not letting Her keep the secret.

ABSTRACT

While they strive to free themselves from musical conventions, many contemporary composers still regard music as a 'language'. However, can innovation be reconciled with communication?

Music seems to convey emotions largely through choreographing listener expectancies. The well-known strong expectancy of leading tone resolving to tonic (the 'attraction' of the former to the latter) can be simulated computationally with a gradient-frequency-neural-network-based artificial neural network (GFNN-ANN). Moreover, GFNN-ANNs seem to have the potential to simulate attractions pertaining to *any musical sound*, thus providing composer with a code for communicating emotions through novel sounds. Consequently, I have implemented a GFNN-ANN, tuned its parameters manually, and tested experimentally how well it fulfills the potential. Athough the results were not particularly satisfactory, GFNN-ANNs are, in my opinion, worth exploring further. However, a faster implementation will be needed.

ABSTRAKT

I když se soudobí skladatelé snaží osvobodit od hudebních konvencí, pro mnoho z nich je hudba stále jakýmsi 'jazykem'. Dá se však inovace sloučit s komunikací?

Zdá se, že hudba povětšinou přenáší emoce tím, že si hraje s očekáváními posluchačů. Dobře známé silné očekávání, že citlivý tón se rozvede do tóniky ('přitažlivost' citlivého tónu k tónice), se dá počítačově simulovat pomocí neuronové sítě založené na tzv. gradient-frequency neural networks (GFNNs). Co víc, vypadá to, že tyto sítě mají potenciál simulovat přitažlivosti spojené s *jakýmkoliv hudebním zvukem*, čímž by skladateli poskytly prostředek pro vyjadřování emocí pomocí nových zvuků. Proto jsem takovou síť naimplementoval, ručně vyladil její parametry a experimentálně ověřil, do jaké míry naplňuje tento potenciál. I když výsledky nebyly příliš uspokojivé, tyto sítě jsou, dle mého názoru, hodny dalšího zkoumání. Bude však potřeba rychlejší implementace.

ACKNOWLEDGMENTS

Many thanks to Ing. Jan Drchal for helping me pursue my favorite subject, to Professor Edward W. Large, Ph.D., for invaluable advice, to Ing. Lukáš Klička for proofreading, to my beloved Hanička for moral support, and to my family for great patience.

CONTENTS

LIST OF FIGURES

ACRONYMS

ANN artificial neural network

GFNN gradient-frequency neural network

GFNN-ANN gradient-frequency-neural-network-based artificial neural network

Network 2 the second gradient-frequency-neural-network-based artificial neural network from [16]

STM software transactional memory

Finally,
my ultimate aim would be to create and master
an entirely personal 'language' [...] which I could use to communicate,
a language which would be as flexible and versatile as, for example,
the musical idioms of the end of the tonal period,
a language that would rediscover
certain universal and permanent categories of musical expression,
without wading through some sort of nostalgia,
or taking one of the 'post-modern' paths
with which we are bombarded today.

— Tristan Murail [24]

1 INTRODUCTION

innovation vs. communication

There is hardly any convention that has stood against the havoc wreaked on Western art music by such composers as Arnold Schoenberg (1874 – 1951) or Luigi Russolo (1883 – 1947). While the music of, e.g., Wolfgang Amadeus Mozart, or, for that matter, that of Red Hot Chili Peppers, is built upon chord progression, Schoenberg's music revolves around tone row and certain transformations of it. Still, not even pitch, traditionally the most important organizing element of Western music, was sacred to Luigi Russolo with his orchestra of self-made noise-generating 'instruments'. And this was only the beginning. In the second half of the 20th century, composers questioned virtually every aspect of Western musical tradition. However, even though they want their music to be unconventional, many contemporary composers still regard it as a means of communication – a 'language'. Now, one would argue that where there is no convention, there also is no language. Indeed, can innovation be reconciled with communication?

GFNN-ANN to reconcile innovation with communication

For some authors, the musical language of Richard Wagner (1813 – 1883) is 'the language of longing'. [13, p. 334] It may not be merely a coincidence that Wagner's common practice was to introduce a chord, which arouses a strong expectancy of resolution into another chord ('attraction'), and then keep listener in suspense by delaying the latter or slap him right away by dispensing with it altogether. [34, 13, pp. 334-339] Indeed, the higher the amount of harmonic expectancy violation, the higher the tension perceived locally and the overall emotional impact of a piece of music. [27] The attraction of leading tone to tonic was simulated computationally with the second gradient-frequency-neural-network-based artificial neural network from [16] (Network 2). Moreover, gradient-frequency-neural-network-based artificial neural networks (GFNN-ANNs) have, I believe, the potential to

simulate attractions pertaining to *any musical sound*, thus providing composer with a code for communicating emotions through novel sounds. Consequently, my thesis is devoted to exploring the possibility of implementing a GFNN-ANN as a utility for contemporary composers that would enable them to test their ideas against this code interactively. I will take Network 2 as a point of departure since it has already shown promising results. The thesis is structured as follows: Chapter 2 will discuss GFNN-ANNs in general, Chapter 3 will recount the adventures of implementing GFNN-ANNs and outline the current implementation, and Chapter 4 will address the following questions experimentally, using the implementation:

1. Can a GFNN-ANN account for attractions of other scale degrees than leading tone to tonic?
2. Does it work for tones (or chords) outside the twelve-tone equal temperament?
3. Can it simulate modulation?

Finally, Chapter 5 will draw conclusions regarding the applicability of GFNN-ANNs to reconciliation of innovation with communication. To facilitate the comprehension of the material by non-musicians, I conclude Chapter 1 with a short introduction to (Western) theory of music, a brief description of MIDI and some basic facts from psychoacoustics.

introduction to (Western) theory of music

Putting radical modernists aside, music can be basically understood as pitches organized in time. All pitches available on standard classical guitar, played on the instrument from the lowest to the highest, four every second, can be notated on grand staff as in Figure 1, or, equivalently, Figure 2. Notice that the interval between adjacent pitches is that of the minor second (semitone). For future reference, the corresponding pitch name and MIDI note number are written below each note. The special symbols after the last note are so-called rests – notated silence. Names differing only in number designate pitches from the same pitch class. Within each pitch class, the pitches are related by the interval of one or more octaves. Pitch classes make up scales. The most common scale is the major scale. The C major and D major scales are notated in Figure 3. Each note is labeled with the corresponding scale degree. Scale degrees have distinct perceptual qualia. For example, leading tone has a strong tendency of resolving to tonic. Notice that the function which maps each pitch class from C major to the pitch class in D major with the same scale degree preserves intervalic relationships within the scale. Actually, each scale (major, minor, chromatic, ...) is characterized by a set of intervalic relationships between its constituent pitch classes. For instance, the chromatic scale has semitones between all adjacent pitch classes. The chromatic scale built on C is notated in Figure 4. Notice that it incorporates all the pitch classes we have seen so far. If I compose a song

using pitches (mostly) from, for example, C major, we say that the song is in the the key of C major. A longer piece of music, like symphony or opera, typically changes key in its course, via so-called modulation. To accompany my song with guitar, I would have to compose a chord progression – a harmony for the song. Each chord in the progression would be built on a scale degree of C major. An example of such progression is given in Figure 5. Notice how each chord inherits the label of the scale degree it is built on. Moreover, perceptual qualia of chords are largely determined by those of its constituent scale degrees. Therefore, dominant chord (which contains leading tone) has a strong tendency of resolving to tonic chord. Also notice that in this particular example each chord takes up exactly one beat. The chords could be made more complex by adding other pitches from the major scale (chords remaining diatonic) or even from the chromatic scale (chords becoming chromatic). A composer like Alois Hába (1893 – 1973) would go still further, incorporating pitch classes in between those of the chromatic scale, thus leaving the realm of the twelve-tone equal temperament we have confined ourselves to so far. A notation for these pitches is shown in Figure 6.

MIDI

MIDI, which stands for Musical Instrument Digital Interface, is a system that allows electronic musical instruments and computers to communicate with each other. For an illustration of its functioning, consider a MIDI keyboard driving a MIDI synthesizer. When you hold down the C4 key on the keyboard, the keyboard sends a C4 note on MIDI message to the synthesizer and the synthesizer plays C4. Then, when you release the key, the keyboard sends a C4 note off message to the synthesizer, and the synthesizer stops playing the C4. In addition to pitch, encoded as MIDI note number with 128 values from 0 to 127 (cf. Figure 1), each note on message transmits information on how hard you have struck the key, encoded as velocity with the same range. [3] Note that for notating instantaneous oscillator frequencies, I used continuous extension of MIDI note number. Also note that *PortMIDI* wraps MIDI messages in events, adding timestamp to each message.

introduction to psychoacoustics

The most important fact of psychoacoustics is that sounds as we hear them are not sounds as they are. Even a single pitch played on a musical instrument usually consists of many sinusoids (pure tones), each with a pitch of its own. Over eons, our auditory system has found it advantageous to fuse the constituent pitches into a single one. In the preceding paragraph, we were concerned with those 'fused' pitches. E.g., if we were to adjust Figure 5 so that it better reflects the acoustical reality, we would have to add to each of the pitches a pitch an octave above, another one a fifth above that, still another one a fourth above that, yet another one a major third above that, and so on.

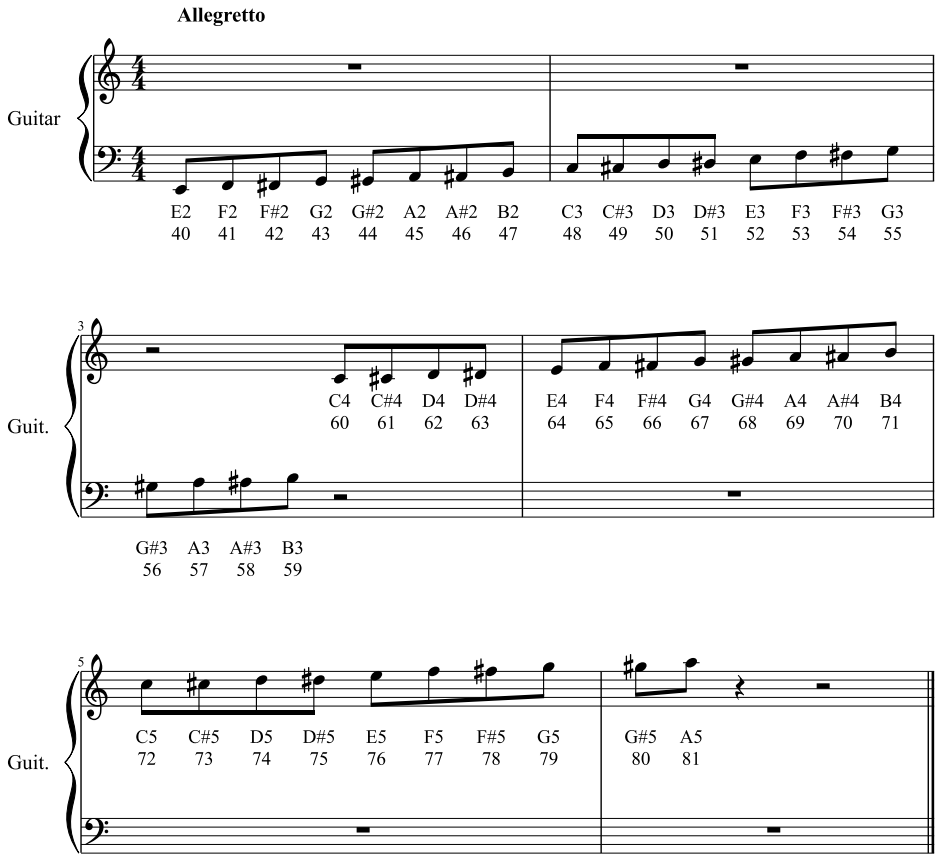

Figure 1: All pitches available on standard classical guitar, played on the instrument from the lowest to the highest, four every second. The interval between adjacent pitches is that of the minor second (semitone). The corresponding pitch name and MIDI note number are written below each note. The special symbols after the last note are so-called rests – notated silence. Names differing only in number designate pitches from the same pitch class. Within each pitch class, the pitches are related by the interval of one or more octaves.

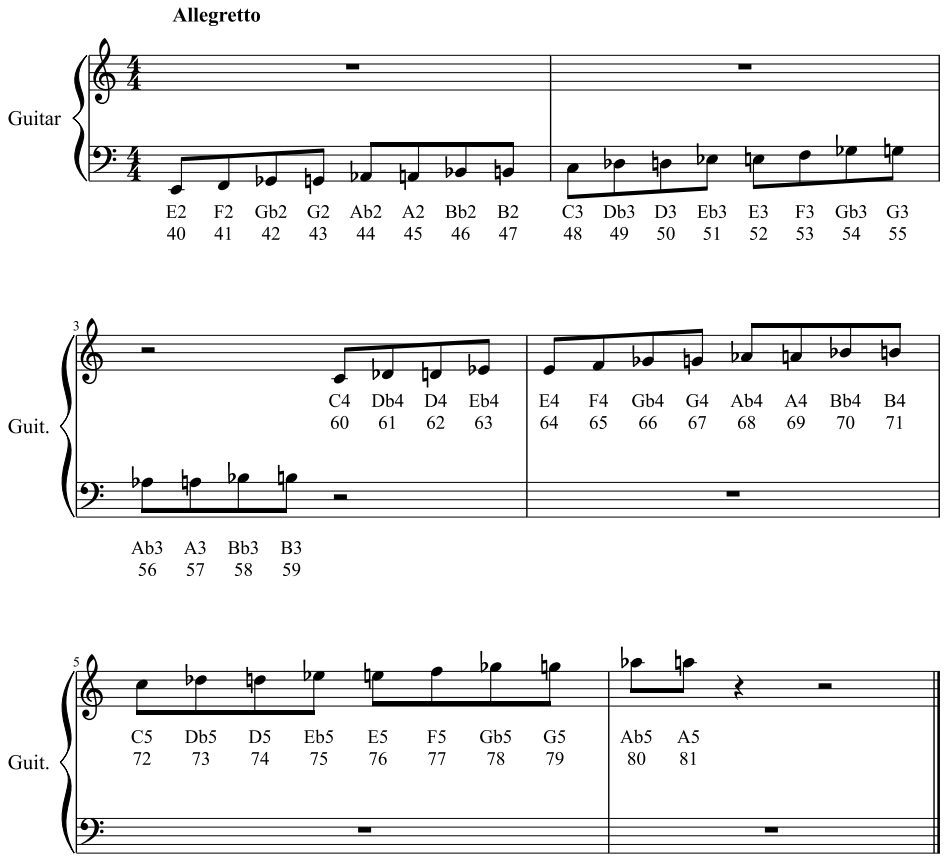

Figure 2: All pitches available on standard classical guitar, played on the instrument from the lowest to the highest, four every second. The interval between adjacent pitches is that of the minor second (semitone). The corresponding pitch name and MIDI note number are written below each note. The special symbols after the last note are so-called rests – notated silence. Names differing only in number designate pitches from the same pitch class. Within each pitch class, the pitches are related by the interval of one or more octaves.

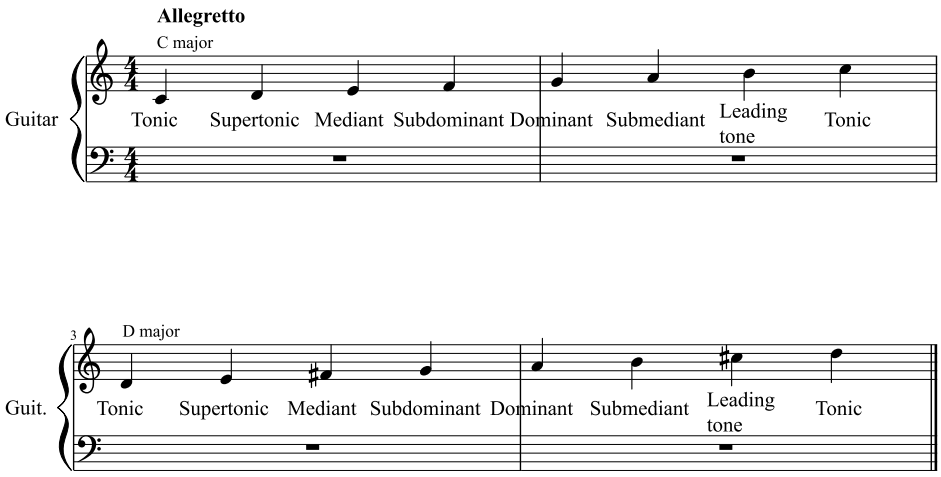

Figure 3: The C major and D major scales. Each note is labeled with the corresponding scale degree. Scale degrees have distinct perceptual qualia. For example, leading tone has a strong tendency of resolving to tonic. The function which maps each pitch class from C major to the pitch class in D major with the same scale degree preserves intervalic relationships within the scale.

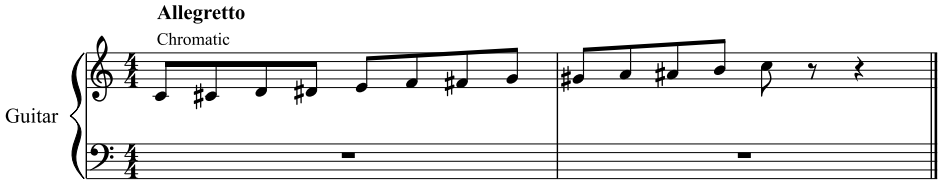

Figure 4: The chromatic scale. It has semitones between all adjacent pitch classes and incorporates all the pitch classes we have seen so far.

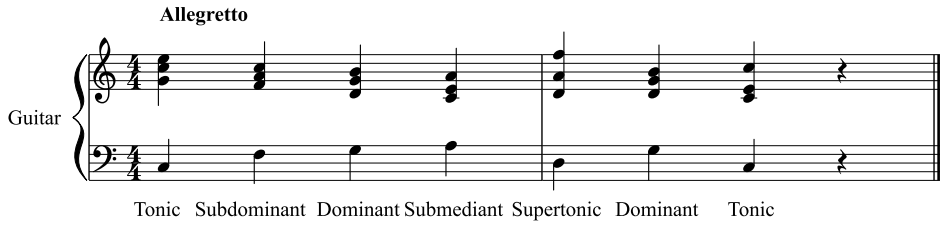

Figure 5: A chord progression in C major. Each chord in the progression is built on a scale degree of C major and inherits its label. In general, perceptual qualia of chords are largely determined by those of its constituent scale degrees. Therefore, dominant chord (which contains leading tone) has a strong tendency of resolving to tonic chord. In this particular example, each chord takes up exactly one beat.

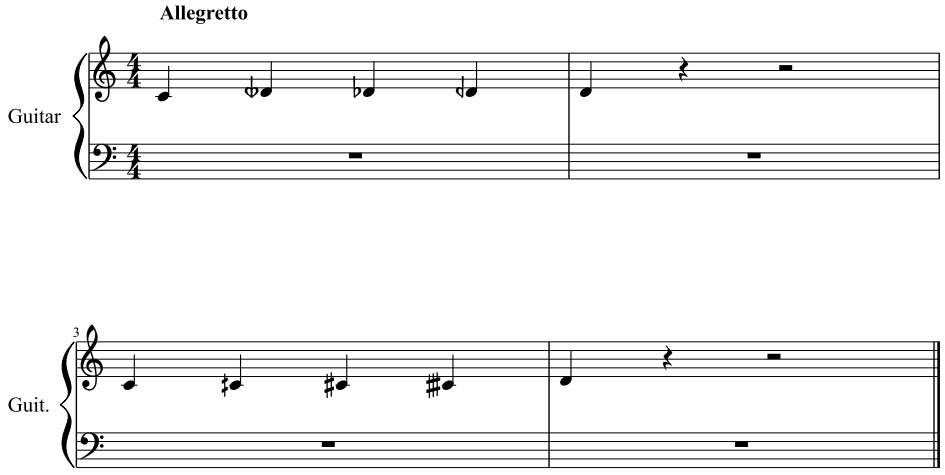

Figure 6: A notation for quarter tones. Top: C4, D4 three quarter tones flat, D4 flat, D4 quarter tone flat, D4. Bottom: C4, C4 quarter tone sharp, C4 sharp, C4 three quarter tones sharp, D4. Both sequences are acoustically equivalent.

There seems to be in us
a sort of affinity to harmonies and rhythms,
which makes some philosophers say that the soul is a harmony,
others, that she possesses harmony.

— Aristotle [8]

2 MODEL

As announced in Chapter 1, this chapter will discuss GFNN-ANNs in general. Starting with description of GFNN-ANN structure, we will then proceed to GFNN-ANN 'neuron'. In both cases, Network 2 will be our point of departure.

GFNN-ANN structure

Network 2 structure is depicted schematically in Figure 7. It comprises two layers of neural oscillators (discussed below). Within each layer, the oscillators are sorted in ascending order by their unique natural frequency, f_j. The frequencies are chosen based on human auditory physiology. Oscillator i may be driven directly by an external acoustic stimulus (music) and/or, through coupling (i, j), by another oscillator, j, in the same or adjacent layer. In general, layers satisfying all the aforementioned properties are called gradient-frequency neural networks (GFNNs). [18] In this thesis, we will designate any artificial neural network (ANN) composed of GFNNs as gradient-frequency-neural-network-based artificial neural network (GFNN-ANN).

GFNN-ANN 'neuron'

GFNN-ANN 'neurons' are so-called neural oscillators. Oscillatory activity of neural systems plays a major role in motor, sensory, and possibly even cognitive functioning. Not only do single pacemaker neurons oscillate, because of their special membrane properties. But large cortical networks oscillate too, via interactions of many excitatory and inhibitory neurons. [9] Neural oscillator is a model of such oscillatory activity. Some better-known neural oscillators are:

1. the Wilson-Cowan oscillator introduced in [33] and
2. the Kuramoto oscillator introduced in [15].

In the following paragraph, we will focus on the neural oscillator underlying Network 2.

Network 2 'neuron'

Let (r_z, ϕ_z) denote the instantaneous amplitude and phase, respectively, of neural oscillator, and (r_x, ϕ_x) the instantaneous amplitude and phase, respectively, of its total input, which takes into account all driving forces from within as well as from outside the GFNN-ANN. The neural oscillator underlying Network 2 is described by the following system of differential equations: [18]

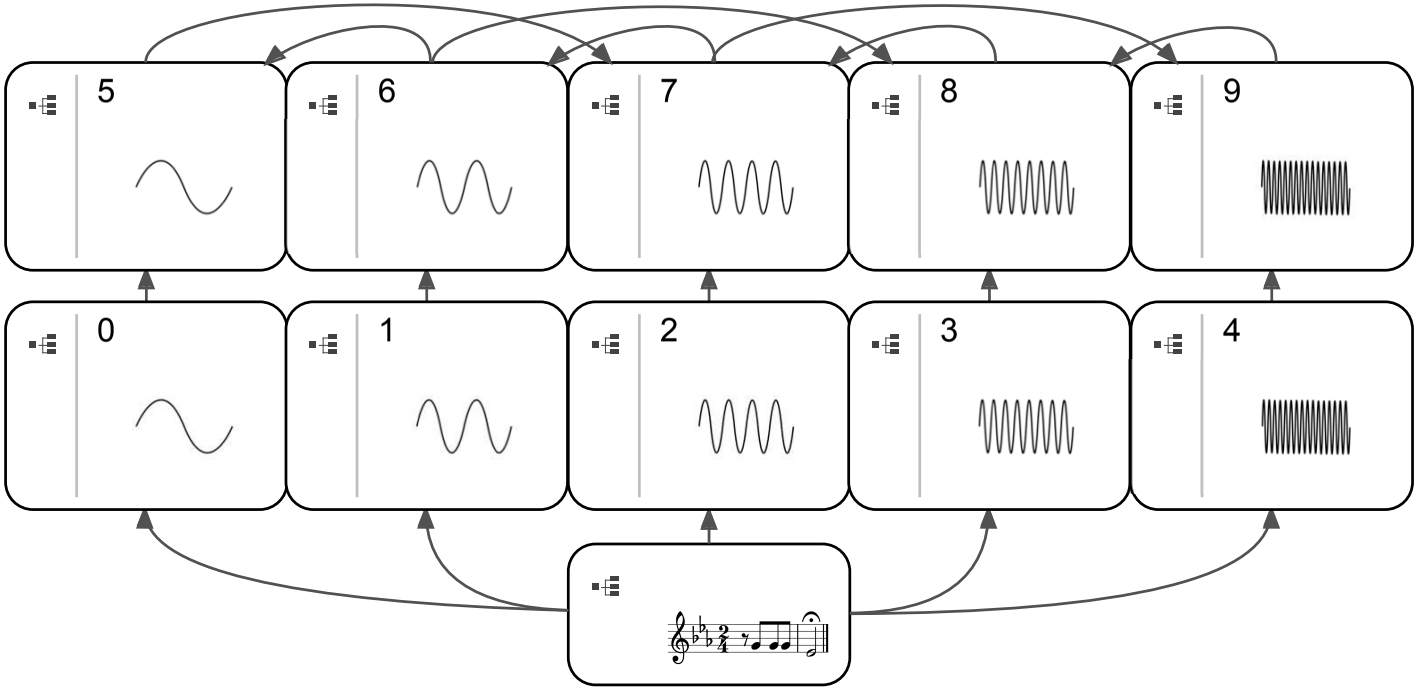

Figure 7: Schematic depiction of Network 2 structure. The external acoustical stimulus (in this case, the opening motif from Beethoven's Fifth) drives all the oscillators in the first layer, which in turn drive the oscillators with corresponding natural frequencies in the second layer, which in turn drive some other oscillators within that layer. Only those oscillators with natural frequencies close to those in the stimulus or related to these by an interval close to the interval of octave and, maybe, even fifth and major third, will oscillate strongly. Their *instantaneous* frequencies will be attracted to the frequencies their *natural* frequencies approximate.

$$
\begin{aligned}
\frac{1}{f}\dot{r_z} &= r_z\left(\alpha + \left(\sum_{k=0}^{p-1}\beta_{k+1}\epsilon^k r_z^{2(k+1)}\right) + \frac{\beta_{p+1}\epsilon^p r_z^{2(p+1)}}{1-\epsilon r_z^2}\right) \\
&\quad + \frac{r_x\left(\epsilon r_x r_z + \cos\left(\phi_z - \phi_x\right)\right) - \sqrt{\epsilon}\left(r_x\cos\phi_z + r_z\cos\phi_x\right)}{\left(1+\epsilon r_x^2 - 2r_x\sqrt{\epsilon}\cos\phi_x\right)\left(1+\epsilon r_z^2 - 2r_z\sqrt{\epsilon}\cos\phi_z\right)}) \\
\frac{1}{f}\dot{\phi_z} &= \omega + \left(\sum_{k=0}^{p-1}\delta_{k+1}\epsilon^k r_z^{2(k+1)}\right) + \frac{\delta_{p+1}\epsilon^p r_z^{2(p+1)}}{1-\epsilon r_z^2} \\
&\quad + \frac{r_x\left(\sin\left(\phi_x - \phi_z\right) + \sqrt{\epsilon}\left(r_x\sin\phi_z - r_z\sin\phi_x\right)\right)}{r_z\left(1+\epsilon r_x^2 - 2r_x\sqrt{\epsilon}\cos\phi_x\right)\left(1+\epsilon r_z^2 - 2r_z\sqrt{\epsilon}\cos\phi_z\right)},
\end{aligned}
\tag{2.1}
$$

where f is natural frequency of the oscillator, ω is angular frequency and α, β_1, ..., β_{p+1}, δ_1, ..., δ_{p+1}, ϵ and p are parameters. Derivation of Equation 2.1 is sumarized in the next paragraph and effects of various settings of the parameters are discussed thereafter.

derivation of Equation 2.1

In this paragraph, I will summarize the derivation of Equation 2.1 by Large et al. [18]. Their point of departure is the following general system of coupled neural oscillators:

$$
\begin{aligned}
\dot{u_i} &= f_i\left(u_i, v_i, \lambda\right) + \epsilon\, p_i\left(u_1, v_1, \ldots, u_n, v_n, \epsilon\right) \\
\dot{v_i} &= g_i\left(u_i, v_i, \lambda\right) + \epsilon\, q_i\left(u_1, v_1, \ldots, u_n, v_n, \epsilon\right),
\end{aligned}
\tag{2.2}
$$

where (u_i, v_i) represents the coordinates of the state of oscillator i, λ represents the set of parameters of the functions f_i and g_i and ϵ is coupling strength. The f_i and g_i functions capture the dependence of u_i and v_i time derivatives, respectively, on the state of oscillator i. The p_i and q_i functions capture the dependence of u_i and v_i time derivatives, respectively, on the states of the oscillators that drive oscillator i. After applying a special transformation to the coordinates and expanding the nonlinear terms[1], we get an equation with a new complex valued state variable, z, resulting from the coordinate transformation:

$$
\dot{z_i} = z_i\left(a_i + b_i \mid z_i \mid^2\right) + x_i + h.o.t., \tag{2.3}
$$

where x is the aggregate input to the oscillator, a and b are complex-valued parameters, and $h.o.t.$ are higher-order terms. The parameters can be related to those from Equation 2.1:

$$
\begin{aligned}
a &= \alpha + \imath\omega, \\
b &= \beta_1 + \imath\delta_1
\end{aligned}
$$

After an expansion of the higher-order terms, we get:

1 To fully grasp this step, one would have to possess knowledge of dynamic systems theory and the Poincaré-Dulac normal form theory.

$$\begin{aligned}\dot{z} &= z\left(a + b_1 \mid z \mid^2 + b_2\epsilon \mid z \mid^4 + b_3\epsilon^2 \mid z \mid^6 + b_4\epsilon^3 \mid z \mid^8 + \ldots\right) \\ &\quad + \left(x + \sqrt{\epsilon}x^2 + \epsilon x^3 + \epsilon\sqrt{\epsilon}x^4 + \ldots\right)\left(1 + \sqrt{\epsilon}\,\bar{z} + \epsilon\bar{z}^2 + \epsilon\sqrt{\epsilon}\,\bar{z}^3 + \ldots\right)\end{aligned} \tag{2.4}$$

Again, the a and b_k parameters can be related to those from Equation 2.1:

$$\begin{aligned}a &= \alpha + \imath\omega, \\ b_k &= \beta_k + \imath\delta_k\end{aligned}$$

Note the infinite geometric series in Equation 2.4. They converge when $\mid z \mid < \frac{1}{\sqrt{\epsilon}}$ and $\mid x \mid < \frac{1}{\sqrt{\epsilon}}$. Under certain conditions on the coefficients of Equation 2.4, the oscillatory dynamics will follow well known cases. For instance, if $b_1 = b$ and $b_k = d$ for other ks, then Andronov–Hopf and Bautin (a.k.a. generalized Hopf) bifurcations are possible (demonstrated in the next paragraph). In both cases the dynamics is oscillatory, which is the regime of interest since we are concerned with neural *oscillators*. The derivation of a closed form expression for Equation 2.4 which can exhibit these bifurcations follows. First, Equation 2.4 is rewritten as:

$$\begin{aligned}\dot{z} &= z\left(a + b_1 \mid z \mid^2\right) + dz\left(\epsilon \mid z \mid^4 + \epsilon^2 \mid z \mid^6 + \epsilon^3 \mid z \mid^8 + \ldots\right) \\ &\quad + \left(x + \sqrt{\epsilon}x^2 + \epsilon x^3 + \epsilon\sqrt{\epsilon}x^4 + \ldots\right)\left(1 + \sqrt{\epsilon}\,\bar{z} + \epsilon\bar{z}^2 + \epsilon\sqrt{\epsilon}\,\bar{z}^3 + \ldots\right)\end{aligned} \tag{2.5}$$

Notice the convergence of the infinite geometric series on the first line of Equation 2.5:

$$\begin{aligned}\epsilon \mid z \mid^4 + \epsilon^2 \mid z \mid^6 + \epsilon^3 \mid z \mid^8 + \ldots &= \epsilon \mid z \mid^4 \left(1 + \epsilon \mid z \mid^2 + \epsilon^2 \mid z \mid^4 + \ldots\right) \\ &= \epsilon \mid z \mid^4 \sum_{k=0}^{\infty} \left(\epsilon \mid z \mid^2\right)^k = \frac{\epsilon \mid z \mid^4}{1 - \epsilon \mid z \mid^2}, \\ \mid z \mid &< \frac{1}{\sqrt{\epsilon}}\end{aligned}$$

Also notice the convergence of the two infinite geometric series on the second line of Equation 2.5:

$$\left(x + \sqrt{\epsilon}x^2 + \epsilon x^3 + \epsilon\sqrt{\epsilon}x^4 + \ldots\right)\left(1 + \sqrt{\epsilon}\,\bar{z} + \epsilon\bar{z}^2 + \epsilon\sqrt{\epsilon}\,\bar{z}^3 + \ldots\right)$$

$$
\begin{aligned}
&= x \sum_{k=0}^{\infty} \left(\sqrt{\epsilon} x\right)^k \sum_{k=0}^{\infty} \left(\sqrt{\epsilon}\, \bar{z}\right)^k \\
&= \frac{x}{1-\sqrt{\epsilon} x} \frac{1}{1-\sqrt{\epsilon}\, \bar{z}}, \\
\mid x \mid &< \frac{1}{\sqrt{\epsilon}}, \\
\mid z \mid &< \frac{1}{\sqrt{\epsilon}}
\end{aligned}
$$

Combining these results, we get:

$$
\begin{aligned}
\dot{z} &= z\left(a + b \mid z \mid^2 + \frac{d\epsilon \mid z \mid^4}{1 - \epsilon \mid z \mid^2}\right) + \frac{x}{1-\sqrt{\epsilon} x} \frac{1}{1-\sqrt{\epsilon}\, \bar{z}}, \\
\mid x \mid &< \frac{1}{\sqrt{\epsilon}}, \\
\mid z \mid &< \frac{1}{\sqrt{\epsilon}}
\end{aligned} \tag{2.6}
$$

Relaxing the assumptions on the coefficients of Equation 2.4, Equation 2.6 can be generalized. This generalized equation will capture more of the dynamic effects caused by the higher order terms present in Equation 2.4. To derive the equation, we must first rewrite Equation 2.4 as:

$$
\begin{aligned}
\dot{z} &= az + z\left(b_1 \mid z \mid^2 + b_2\epsilon \mid z \mid^4 + \ldots + b_p \epsilon^{p-1} \mid z \mid^{2p}\right) \\
&\quad + z\left(b_{p+1}\epsilon^p \mid z \mid^{2(p+1)} + b_{p+2}\epsilon^{p+1} \mid z \mid^{2(p+2)} + \ldots\right) \\
&\quad + \left(x + \sqrt{\epsilon} x^2 + \epsilon x^3 + \epsilon\sqrt{\epsilon} x^4 + \ldots\right)\left(1 + \sqrt{\epsilon}\, \bar{z} + \epsilon \bar{z}^2 + \epsilon\sqrt{\epsilon}\, \bar{z}^3 + \ldots\right) \\
&= az + z \sum_{k=0}^{p-1} b_{k+1} \epsilon^k \mid z \mid^{2(k+1)} \\
&\quad + z b_{p+1} \sum_{k=p}^{\infty} \epsilon^k \mid z \mid^{2(k+1)} \\
&\quad + \left(x + \sqrt{\epsilon} x^2 + \epsilon x^3 + \epsilon\sqrt{\epsilon} x^4 + \ldots\right)\left(1 + \sqrt{\epsilon}\, \bar{z} + \epsilon \bar{z}^2 + \epsilon\sqrt{\epsilon}\, \bar{z}^3 + \ldots\right)
\end{aligned} \tag{2.7}
$$

Here, the last parameter from Equation 2.1, p, finally occurs. Note that for $k = p$, $\epsilon^p \mid z \mid^{2(p+1)}$ is a factor of $\sum_{k=p}^{\infty} \epsilon^k \mid z \mid^{2(k+1)}$ so the infinite geometric series on the second line of Equation 2.7 can be written as:

$$
z b_{p+1} \epsilon^p \mid z \mid^{2(p+1)} \sum_{k=p}^{\infty} \epsilon^{k-p} \mid z \mid^{2(k-p)}
$$

Let $m = k - p$ which implies that for $k = p$, $m = 0$ and for $k = \infty$, $m = \infty - p = \infty$. Then, the series can be written as:

$$z b_{p+1} \epsilon^p \mid z \mid^{2(p+1)} \sum_{m=0}^{\infty} \left(\epsilon \mid z \mid^2\right)^m$$

Note the convergence of the series:

$$\begin{aligned} z b_{p+1} \epsilon^p \mid z \mid^{2(p+1)} \sum_{m=0}^{\infty} \left(\epsilon \mid z \mid^2\right)^m &= z \frac{b_{p+1} \epsilon^p \mid z \mid^{2(p+1)}}{1 - \epsilon \mid z \mid^2}, \\ \mid z \mid &< \frac{1}{\sqrt{\epsilon}} \end{aligned}$$

Putting it all together with the previous result for the two infinite geometric series on the third line of Equation 2.7, we obtain:

$$\begin{aligned} \dot{z} &= z \left(a + \left(\sum_{k=0}^{p-1} b_{k+1} \epsilon^k \mid z \mid^{2(k+1)} \right) + \frac{b_{p+1} \epsilon^p \mid z \mid^{2(p+1)}}{1 - \epsilon \mid z \mid^2} \right) \\ &\quad + \frac{x}{1 - \sqrt{\epsilon} x} \frac{1}{1 - \sqrt{\epsilon} \bar{z}}, \\ \mid x \mid &< \frac{1}{\sqrt{\epsilon}}, \\ \mid z \mid &< \frac{1}{\sqrt{\epsilon}} \end{aligned} \tag{2.8}$$

We can write Equation 2.8 in polar form using polar representations for x and z:

$$\begin{aligned} x &= r_x e^{\imath \phi_x}, \\ z &= r_z e^{\imath \phi_z}, \\ \bar{z} &= r_z e^{-\imath \phi_z}, \\ \dot{z} &= e^{\imath \phi_z} \left(\dot{r}_z + \imath r_z \dot{\phi}_z \right) \end{aligned}$$

Also, we relate the a and b_k parameters to those of Equation 2.1 in the same way as before:

$$\begin{aligned} a &= \alpha + \imath \omega, \\ b_k &= \beta_k + \imath \delta_k, \end{aligned}$$

By substituting into Equation 2.8 using Euler's formula

$$r e^{\imath \phi} = r \left(cos\left(\phi\right) + \imath sin\left(\phi\right) \right),$$

equating real and imaginary parts of the left and right hand sides of Equation 2.8, and multiplying right hand sides of both resulting equations by natural oscillator frequency, f, we obtain Equation 2.1.

Equation 2.1 parameter settings

Depending on setting of the α, β_1, ..., β_{p+1}, δ_1, ..., δ_{p+1}, ϵ and p parameters in Equation 2.1, the oscillator exhibits various behaviors. Generally, if $\delta_k = 0$ for all ks, the instantaneous frequency of the oscillator is independent of its instantaneous amplitude. If, on the other hand, $\delta_k \neq 0$ for some k, the frequency is dependent on the amplitude. Also in general, the higher the value of ϵ, the more nonlinear the oscillator is. For some settings satisfying $\alpha \leq 0$ and $\beta_k < 0$ for all ks, the oscillator begins to oscillate only after a stimulus has been switched on and decays after it has been switched off – the oscillator undergoes a supercritical Hopf bifurcation (see Figure 8). For some other settings, satisfying $\alpha < 0$, $\beta_1 > 0$, $\beta_k < 0$ for other ks, the oscillator also begins to oscillate after a stimulus has been switched, but, instead of decaying after it has been switched off, continues oscillating with the stimulus frequency – the oscillator undergoes Bautin bifurcation (see Figure 9). Finally, for $\alpha > 0$, the oscillator may oscillate spontaneously (see Figure 10). These behaviors are especially relevant to modeling music-induced expectancies. [16, 17]

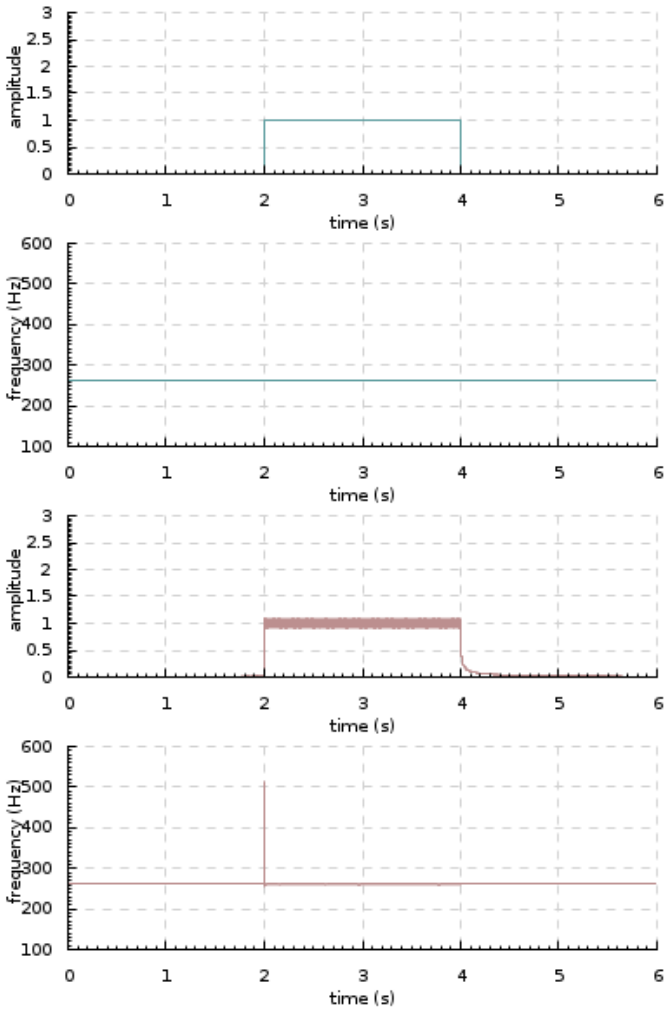

Figure 8: Neural oscillator from Equation 2.1 with $f = 261.6255653006\,Hz$, $\alpha = 0$, $\omega = 2\pi$, $\beta_1 = \ldots = \beta_{p+1} = -1$, $\delta_1 = \ldots = \delta_{p+1} = 0$, $\epsilon = 0.1$ and $p = 1$ (rosybrown) stimulated by pure tone with the same frequency (cadetblue). The oscillator begins to oscillate only after the stimulus has been switched on and decays after it has been switched off – the oscillator undergoes a supercritical Hopf bifurcation.

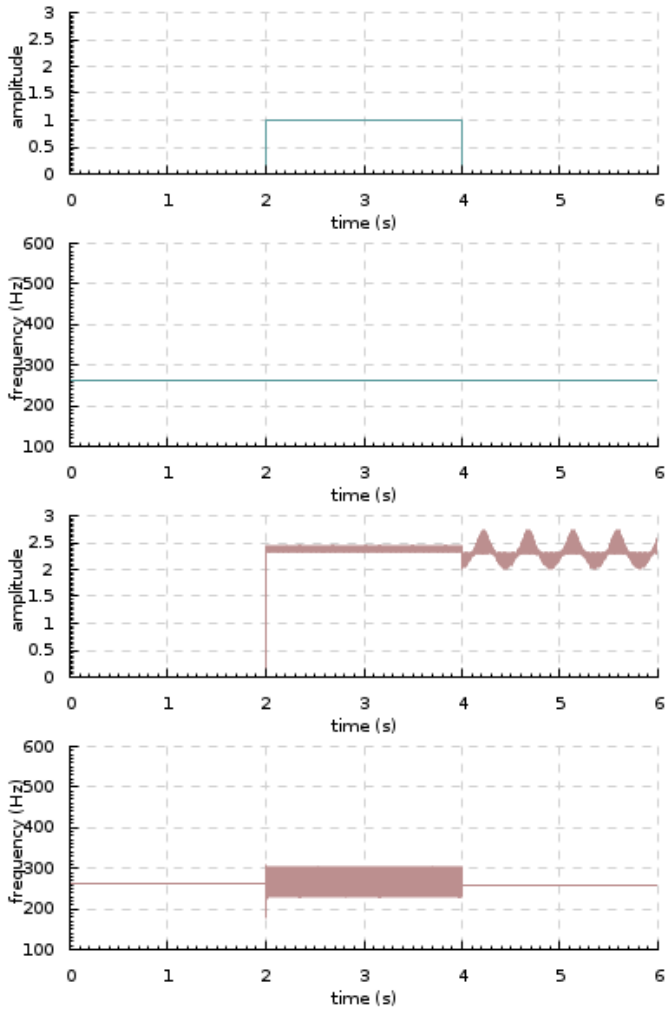

Figure 9: Neural oscillator from Equation 2.1 with $f = 261.6255653006\,Hz$, $\alpha = -0.4$, $\omega = 2\pi$, $\beta_1 = 1.2$, $\beta_2 = \ldots = \beta_{p+1} = -1$, $\delta_1 = -0.01$, $\delta_2 = \ldots = \delta_{p+1} = 0$, $\epsilon = 0.1$ and $p = 1$ (rosybrown) stimulated by pure tone with the same frequency (cadetblue). The oscillator begins to oscillate only after the stimulus has been switched and continues oscillating with the stimulus frequency after it has been switched off – the oscillator undergoes Bautin bifurcation.

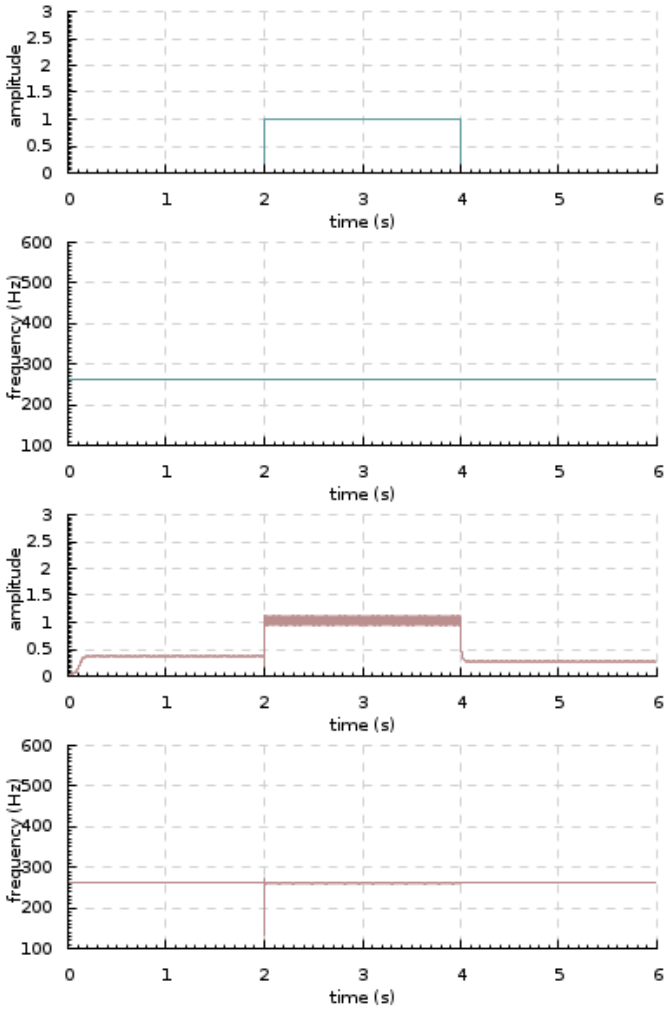

Figure 10: Neural oscillator from Equation 2.1 with $f = 261.6255653006\,Hz$, $\alpha = 0.1$, $\omega = 2\pi$, $\beta_1 = \ldots = \beta_{p+1} = -1$, $\delta_1 = \ldots = \delta_{p+1} = 0$, $\epsilon = 0.1$ and $p = 1$ (rosybrown) stimulated by pure tone with the same frequency (cadetblue). The oscillator oscillates spontaneously.

[...]
Oh, infinite progeny of Church, Hope, and ML,
I curry favor not when I say:
Scan me right, fold me left,
Lazy lady of many shapes, you've got class.

— Don Smith, A Haskell Lover's Plea [26]

3 IMPLEMENTATION

As promised in Chapter 1, this chapter will recount the adventures of implementing GFNN-ANNs and outline the current implementation. But first, I will clarify the requirements that drove the implementation process.

requirements

Inasmuch as I was interested in developing an *interactive* utility, I wanted the implementation to be as fast as possible. At the same time, I wanted it to be sufficiently generic and modular, so that, in the future, I can easily plug a different GFNN-ANN for modeling sound-related expectancies into it or even add one for modeling rhythm-related expectancies. In fact, we could simply take Network 2, reduce the natural oscillator frequencies, make some minor adjustments to the parameter settings, and model perception of beat in highly syncopated music (like salsa) with it right away (see [31]). Last but not least, I wanted to program in my beloved language – Haskell. Since the reader most likely has not heard of Haskell yet, a short characterization of the language follows in the next paragraph.

Haskell

Instead of trying to describe Haskell in one paragraph myself, I will paraphrase a characterization by Lipovaca [20], which is, in my opinion, concise and fitting. Haskell is a purely functional language, it is lazy, statically typed, and elegant and concise. Elaborating on the first point, in imperative language, the program is a sequence of tasks to be executed by computer. On the other hand, in purely functional language, the program is a set of declarations. Quoting Lipovaca's example, *the factorial of a number is the product of all the numbers from 1 to that number, the sum of a list of numbers is the first number plus the sum of all the other numbers, and so on*. The building blocks of the declarations are functions with no side-effects. The only thing a function can do is calculate something based on the inputs and return it as the output[1]. Consequently, if a function is called twice with the same inputs, it's guaranteed to give the same output. Not only does this

1 To make the reader's head explode: pure function can return an action that modifies a variable or another action which launches nuclear missiles on Mongolia and then returns an action that makes Bugs Bunny jump about and sing the Marseillaise.

allow the compiler to reason about the program's behavior (and, e.g., make agressive optimizations), but it also allows the programmer to see whether a function is correct or not clearly and then build more complex functions by putting simple functions together without the fear that the simple functions will fail in the new context. The second point, laziness, means that unless specifically told otherwise, Haskell won't calculate values until they are absolutely needed. Conceptually, this reduces any program to a series of transformations on data. Due to the statical typing, in Haskell, a great deal of possible errors is caught at compile time. Finally, since Haskell is built around abstractions, it is elegant and concise – Haskell programs are usually shorter than their imperative equivalents.

adventure with actors

Now, let's turn to implementing GFNN-ANNs. The first question that needed to be resolved was how to represent GFNN-ANNs. I decided to use the *actor* package by Sulzmann [28] which implements the actor model of concurrent computation [1]. The model consists of computational agents (actors) communicating through asynchronous message passing. More specifically, actor is computational agent which maps each incoming communication to a 3-tuple consisting of:

1. a finite set of communications sent to other actors
2. a new behavior (which will govern the response to the next communication processed), and
3. a finite set of new actors created

Languages implementing the actor model, such as Erlang, commonly only support single-headed pattern matching over received messages. That is, each receive pattern[2] matches at most one message. In the *actor* package, the Erlang style actors are extended with receive clauses containing multi-headed message patterns. This means that one receive pattern can match multiple messages. [29] For the GFNN-ANNs, my idea was to represent each neural oscillator with an actor receiving samples from an external acoustic source and/or couplings with driving oscillators and sending its samples to couplings with driven oscillators. Correspondigly, each coupling would be represented with an actor receiving samples from the driving oscillator and sending them modified (typically with the amplitude reduced) to the driven oscillator. However, for the utility to be useful to composers, it must have a way to take snapshots of the entire GFNN-ANN, and this proved to be prohibitively hard with this representation. Consequently, the implementation based on this representation was abandoned unfinished.

adventure with CUDA

Next, I tried using the *accelerate* package by Chakravarty et al. [4] and the *accelerate-cuda* package by Chakravarty et al. [5]. The *accelerate* package defines *Accelerate* – an embedded domain-specific language

2 For imperative programmers: a condition in an if statement

of parameterised, collective array computations. The *accelerate-cuda* package provides a CUDA backend which runs *Accelerate* computations as follows: initially, it generates GPU device code which implements the computation, while simultaneously transferring the input arrays to the device. Next, it executes the various GPU kernels and manages intermediate storage on the device. Finally, the resulting arrays are transferred back from the device to the host. All this happens at application runtime. [6] The core of GFNN-ANN representation consisted of five *Accelerate* vectors. Network 2 in this representation is depicted schematically in Figure 11. Unfortunately, *Accelerate* has its own numeric types and this complicated interfacing with other packages. Moreover, solving Equation 2.1 for a GFNN-ANN using the CUDA backend in a loop resulted in *Service Temporarily Unavailable* Mac OS X errors. As a result, in the current representation, *Accelerate* vectors have been replaced by unboxed vectors from the *vector* package by Leshchinskiy [19]. The latter enable storing simple, atomic types, and pair types in consecutive memory slots, without pointers. [11] Moreover, they support in-place updates – a feature exploited in performance-critical parts of the current implementation.

r_{z_i}	ϕ_{z_i}	$indeg_{z_i}$	$r_{c_{ij}}$	$\phi_{c_{ij}}$
r_{z_0}	ϕ_{z_0}	0	$r_{c_{50}}$	$\phi_{c_{50}}$
r_{z_1}	ϕ_{z_1}	0	$r_{c_{56}}$	$\phi_{c_{56}}$
r_{z_2}	ϕ_{z_2}	0	$r_{c_{61}}$	$\phi_{c_{61}}$
r_{z_3}	ϕ_{z_3}	0	$r_{c_{67}}$	$\phi_{c_{67}}$
r_{z_4}	ϕ_{z_4}	0	$r_{c_{72}}$	$\phi_{c_{72}}$
r_{z_5}	ϕ_{z_5}	2	$r_{c_{75}}$	$\phi_{c_{75}}$
r_{z_6}	ϕ_{z_6}	2	$r_{c_{78}}$	$\phi_{c_{78}}$
r_{z_7}	ϕ_{z_7}	3	$r_{c_{83}}$	$\phi_{c_{83}}$
r_{z_8}	ϕ_{z_8}	3	$r_{c_{86}}$	$\phi_{c_{86}}$
r_{z_9}	ϕ_{z_9}	2	$r_{c_{89}}$	$\phi_{c_{89}}$
			$r_{c_{94}}$	$\phi_{c_{94}}$
			$r_{c_{97}}$	$\phi_{c_{97}}$

Figure 11: Schematic depiction of Network 2 in the *Accelerate*-based representation (cf. Figure 12 below). r_{z_i} and ϕ_{z_i} stand for instantaneous amplitude and phase of oscillator i, respectively, $indeg_i$ for number of oscillators driving oscillator i, and $r_{c_{ij}}$ and $\phi_{c_{ij}}$ for amplitude and phase, respectively, of the coupling between driven oscillator i and driving oscillator j.

adventure with signal functions

In parallel with tweaking the vector-based GFNN-ANN representation, I strived to find a suitable architecture for the app. The *Euterpea* package by Hudak [12] seemed to be an especially elegant solution. It defines an embedded domain-specific language of signal functions

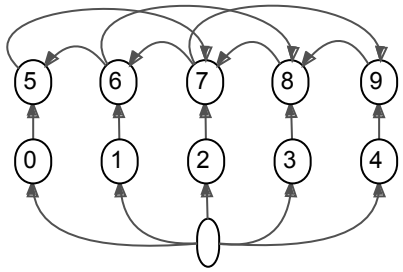

Figure 12: Network 2 connectivity.

for computer music development. Whereas vanilla functions operate on values, signal functions operate on time-varying signals. Programming in *Euterpea* typically involves taking a bunch of signal functions, either predefined or lifted from vanilla functions, lifting being a no-brainer, and then connect an output of one to an input of another with arrow (literally). This allows for a great deal of modularity. E.g., I had a signal function transforming real-time MIDI input to an acoustic signal, another solving Equation 2.1 for the GFNN-ANN each time a sample of the acoustic signal arrives and having a signal of the solutions as the output, and still another rendering the signal of solutions as video. However, this implementation suffered from massive memory leaks I was not able to plug. Even the examples included in the package turned out to be memory hogs. Therefore, in the current implementation, I have resorted to a combination of explicit concurrency and software transactional memory (STM). Since STM is a relatively new approach to programming shared-memory parallel processors, I will devote the next paragraph to it.

STM

The key idea of STM is that a block of code can be enclosed by an atomic block which is guaranteed to run atomically with respect to every other atomic block. An atomic block does not take locks on the accessed variables. Instead, it maintains a log of all memory reads and writes it makes. When the block completes, it checks whether any of the variables it has read, according to the log, was modified by another thread during the block's execution. If some of them was, the block is re-executed from scratch. If none of them was, writes from the log are commited to memory. Transactional memory eliminates many pitfalls of lock-based programming. For instance, since there are no locks, there are no lock-induced deadlocks. There is no priority inversion, either. [10]

current implementation

The architecture of the current implementation is depicted in Figure 13. The system is composed of five independent modules, each running in a separate thread (the first four in light threads [14] and the last in the main thread) and communicating with others via bounded transacted channels [30]. It works roughly as follows: at regular time intervals, *MIDI Receiver* polls a MIDI input device for events, using *PortMIDI* [21], and sends the events to *Input Generator.* Based on the MIDI events received so far, *Input Generator* generates audio sam-

ples and sends the samples to GFNN *Simulator* (see below for details). For each audio sample, GFNN *Simulator* solves Equation 2.1 for the GFNN-ANN and sends the solution to GFNN *Renderer* (see below for details). GFNN *Renderer* renders each solution as graphics, using *OpenGL* [7], and sends the graphics to GFNN *Canvas*. Finally, GFNN *Canvas* displays the graphics on the screen, using *GLFW* [22].

Figure 13: The architecture of the current implementation. The system is composed of five independent modules, each running in a separate thread and communicating with others via shared variables. At regular time intervals, *MIDI Receiver* polls a MIDI input device for events and sends the events to *Input Generator.* Based on the MIDI events received so far, *Input Generator* generates audio samples and sends the samples to GFNN *Simulator*. For each audio sample, GFNN *Simulator* solves Equation 2.1 for the GFNN-ANN and sends the solution to GFNN *Renderer*. GFNN *Renderer* renders each solution as graphics and sends the graphics to GFNN *Canvas*. Finally, GFNN *Canvas* displays the graphics on the screen.

Input Generator

Internally, the MIDI events passed to *Input Generator* drive a bank of sinusoidal oscillators whose basic representation can be seen in Figure 14. Conceptually, each amplitude envelope in the *envs* vector is represented with the following pair of functions (cf. Figure 14, Figure 16):

$$g_{on}(t_{sim}) = \begin{cases} 0 & t_{sim} \leq t_{on} \\ \dot{r}\,(t_{sim} - t_{on}\ ) & t_{sim} \leq \frac{r_{max}+\dot{r}t_{on}}{\dot{r}} \\ r_{max} & \end{cases} \tag{3.1}$$

$$g_{off}(t_{sim}) = \begin{cases} 0 & t_{sim} \leq t_{off} \\ -\dot{r}\,(t_{sim} - t_{off}) & t_{sim} \leq \frac{r_{max}+\dot{r}t_{off}}{\dot{r}} \\ -r_{max} & \end{cases}$$

where t_{sim} is a simulation time instant, t_{on} the timestamp from the corresponding note on MIDI event, t_{off} the timestamp from the corresponding note off MIDI event, $\dot{r}$ the time derivative of instantaneous oscillator amplitude between t_{on} and $\frac{r_{max}+\dot{r}t_{on}}{\dot{r}}$, the instant it reaches r_{max}, $-\dot{r}$ the time derivative of instantaneous oscillator amplitude between t_{off} and $\frac{r_{max}+\dot{r}t_{off}}{\dot{r}}$, the instant it drops to 0, and r_{max} the oscilla-

$inputs\,(i)$ is updated by adding the sample. Additionally, the $inputs$ vector is updated as specified by the following formula:

$$r_{x_i}\left(\cos\phi_{x_i}+\imath\sin\phi_{x_i}\right) \quad += \quad \sum_j r_{c_{ij}} r_{z_j}\left(\cos\left(\phi_{c_{ij}}+\phi_{z_j}\right)+\imath\sin\left(\phi_{c_{ij}}+\phi_{z_j}\right)\right), \tag{3.4}$$

where (r_{x_i}, ϕ_{x_i}) is instantaneous amplitude and phase, respectively, of the aggregate input to oscillator i, $\left(r_{c_{ij}}, \phi_{c_{ij}}\right)$ amplitude and phase, respectively, of the connection between driving oscillator j and driven oscillator i, and $\left(r_{z_j}, \phi_{z_j}\right)$ instantaneous amplitude and phase, respectively, of oscillator j. Next, Equation 2.1 is solved numerically for the GFNN-ANN using Euler's method. In other words, the $\dot{zs}$ vector is updated using Equation 2.1 and the zs vector is updated with the following formula, for a time step dt:

$$\begin{aligned} r_{z_i} \quad &+= \quad dt\ \dot{r}_{z_i} \\ \phi_{z_i} \quad &+= \quad dt\ \dot{\phi}_{z_i} \end{aligned} \tag{3.5}$$

Finally, after some additional processing, the updated zs and $\dot{zs}$ vectors are sent to GFNN *Renderer*.

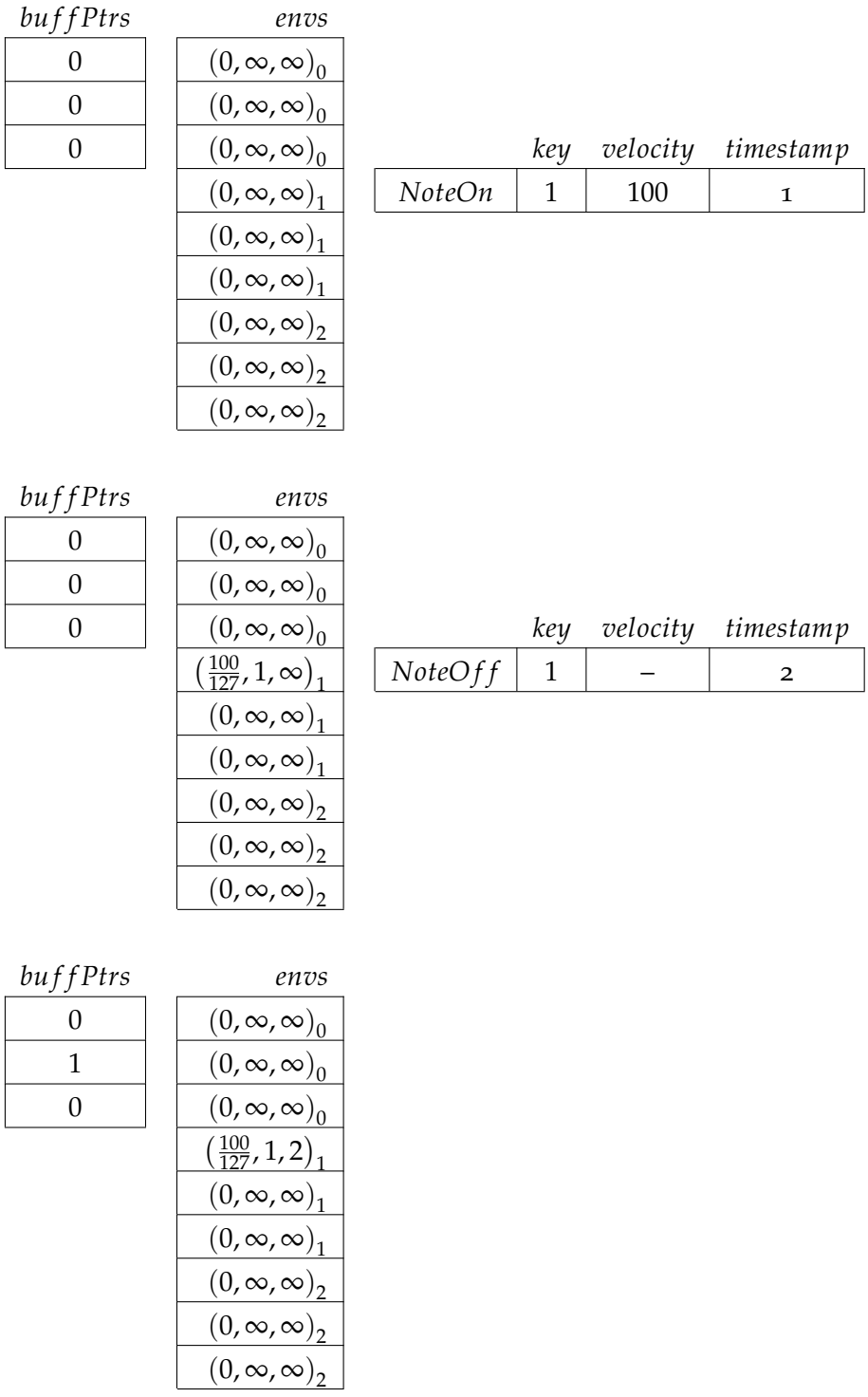

Figure 15: An example of MIDI note on/off event updating the *envs* and *buffPtrs* vectors.

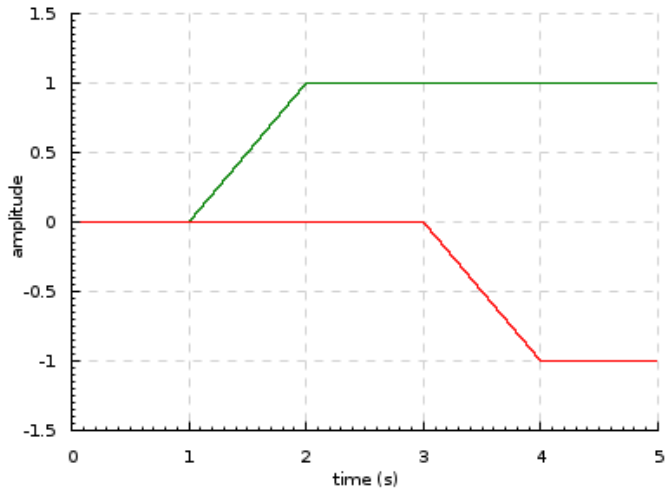

Figure 16: Amplitude envelope with $\dot{r} = 1$, $r_{max} = 1$, $t_{on} = 1\,s$ and $t_{off} = 3\,s$ as represented, conceptually, inside *Input Generator*. $\dot{r}$ is the time derivative of instantaneous oscillator amplitude between t_{on} and $\frac{r_{max}+\dot{r}t_{on}}{\dot{r}}$, the instant it reaches r_{max}, $-\dot{r}$ the time derivative of instantaneous oscillator amplitude between t_{off} and $\frac{r_{max}+\dot{r}t_{off}}{\dot{r}}$, the instant it drops to 0, r_{max} the oscillator amplitude reflecting the velocity from the corresponding note on MIDI message, t_{on} the timestamp from the corresponding note on MIDI event, and t_{off} the timestamp from the corresponding note off MIDI event. The green plot corresponds to g_{on} and the red one to g_{off}.

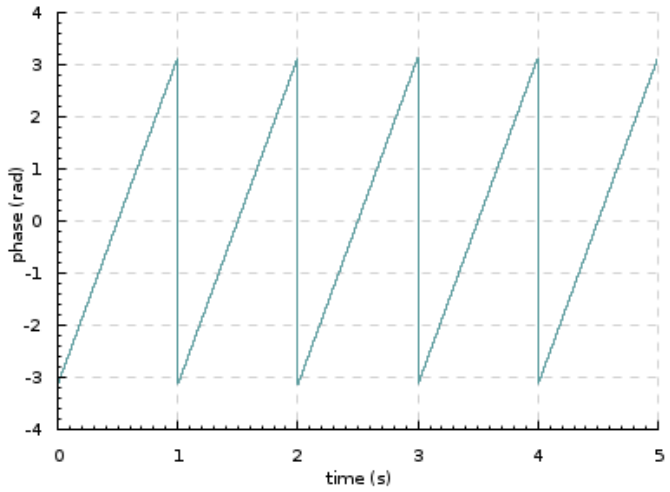

Figure 17: Phase of sinusoidal oscillator with $f = 1Hz$. The function for instantaneous oscillator phase was devised so that its range is $(-\pi, pi]$, the default range in the Haskell package for manipulating complex numbers.

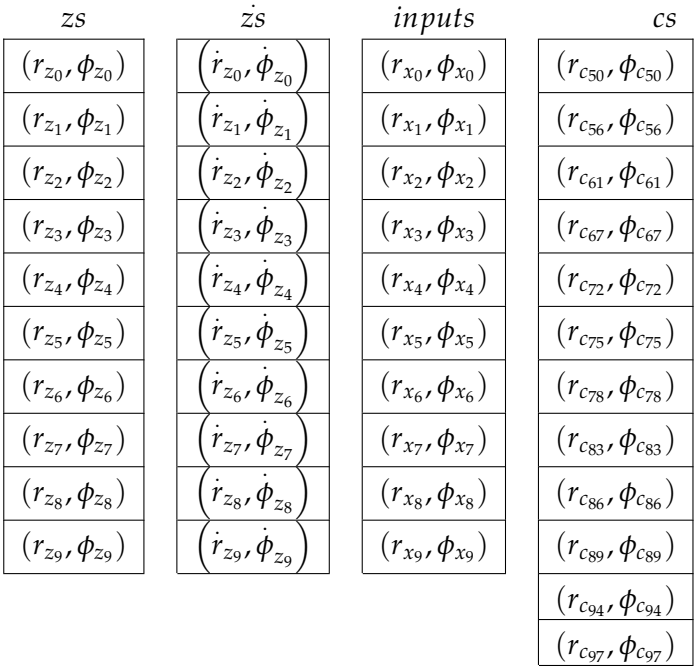

Figure 18: Network 2 as it would be, in essence, represented in GFNN *Simulator* (cf. Figure 19 below). r_{z_i} and ϕ_{z_i} stand for instantaneous amplitude and phase of oscillator i, respectively, r_{x_i} and ϕ_{x_i} for instantaneous amplitude and phase, respectively, of the aggregate input to oscillator i, and $r_{c_{ij}}$ and $\phi_{c_{ij}}$ for amplitude and phase, respectively, of the coupling between driven oscillator i and driving oscillator j.

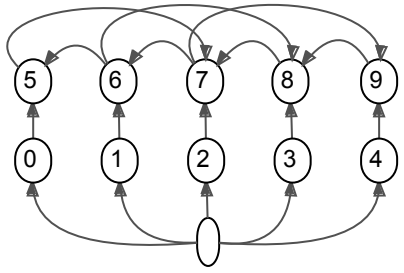

Figure 19: Network 2 connectivity.

[...] tonality itself –
with its process of instilling expectations
and subsequently witholding promised fulfillment until climax –
is the principal musical means during the period from 1600 to 1900
for arousing and channeling desire.

— Susan McClary [23]

4 EVALUATION

As stated in Chapter 1, this chapter will address the following questions experimentally, using the implementation outlined in Chapter 3:

1. Can a GFNN-ANN account for attractions of other scale degrees than leading tone to tonic[1]?
2. Does it work for tones (or chords) outside the twelve-tone equal temperament?
3. Can it simulate modulation[2]?

First, tuning of the GFNN-ANN for the experiments will be summarized. Next, four experiments will be described, the first two addressing the first question and the latter two the rest. Finally, the results will be discussed.

tuning GFNN-ANN

As mentioned in Chapter 1, Network 2 served as a point of departure. I was able to obtain most of the parameter settings from the paper or directly from Prof. Large[3]. For the rest, I resorted to trial and error (the approach reportedly taken even by Prof. Large himself for the paper). This mainly concerned exact connection strengths. For connections within the first layer, I came up with the following Gaussian kernel:

$$r_{c_{ij}} = 0.01 \frac{1}{2\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{|pitch_i - pitch_j|}{2}\right)^2}, \tag{4.1}$$

where $r_{c_{ij}}$ is strength of the connection from oscillator j to oscillator i and $pitch_i$ natural frequency of oscillator i expressed as MIDI note number[4]. For inter-layer connections, I used the uniform strength

1 In C major, leading tone is B, one semitone below tonic, C, in D major, leading tone is C#, one semitone below tonic, D, and so on.
2 Change of key, for example, from G major to C major.
3 Email correspondence
4 More precisely, the continuous extension of it.

$$r_{c_{ij}} = \begin{cases} \frac{1}{\sqrt{10}} & pitch_i = pitch_j \\ 0 \end{cases} \tag{4.2}$$

Within the second layer, I replaced the original major-scale-based connectivity with one based on the chromatic scale, specified by the following formula:

$$r_{c_{ij}} = \begin{cases} \frac{1}{\sqrt{500}} & \mid pitch_i - pitch_j \mid \leq 1.55 \\ \frac{1}{\sqrt{500}} & \mid \left[(pitch_i - pitch_j) \mod 12 \right] - nst \mid \leq 0.05 \\ 0 \end{cases} \tag{4.3}$$

where *nst* stands for the nearest interval from the chromatic scale. The first case of Equation 4.3 enables connections between oscillators with close natural frequencies. Due to this, an oscillator can attract the instantaneous frequencies of other oscillators with close natural frequencies to its own instantaneous frequency. The second case enables connections between oscillators whose natural frequencies correspond to pitch classes[5] related by an interval close to an interval from the chromatic scale. I hoped this would help in modeling attractions of chromatic chords[6] to tonic. Overall, I strived to tune the network so that, when stimulated with the C major chord[7], the corresponding pitches remain salient in the second layer even after the stimulus is gone and the first layer becomes silent. According to Prof. Large, this is a sufficient condition for attraction to take place. Since the stimulus decay in the first layer was too slow, I decreased α parameter in this layer, yielding the following settings of oscillators in the network: $\alpha = -0.1$, $p = 2$, $\beta_1 = \beta_2 = -1$, $\delta_1 = \delta_2 = 0$, $\epsilon = 0.1$ for the first layer, $\alpha = -0.4$, $p = 2$, $\beta_1 = 1.2$, $\beta_2 = -1$, $\delta_1 = 0.01$, $\delta_2 = 0$, $\epsilon = 0.78$ for the second, and $\omega = 2\pi$ for both.

testing GFNN-ANN for attractions to tonic

The first three experiments partially replicate an experiment by Woolhouse and Cross [34], with the GFNN-ANN in place of human subjects. In the original study, subjects were asked to rate the degree of attraction and/or resolution they felt from one chord to another and from a chord to a single pitch. Within each pair, one stimulus followed the other without a break. The chords were paired with each other in both temporal orders, the single pitch, however, always followed the chord. The stimuli, each two seconds long, were presented using tones consisting of superposed octave-related sinusoids with overall amplitude controlled by a Gaussian function. This technique

5 For instance, ..., C3, C4, C5, ... pitches all belong to the same pitch class, C, and are related by integral multiples of the interval of octave, which spans 12 semitones. This is why there is the modulo 12 in Equation 4.3.

6 That is, those containing some chromatic scale degrees.

7 A chord consisting of C, E and G pitch classes.

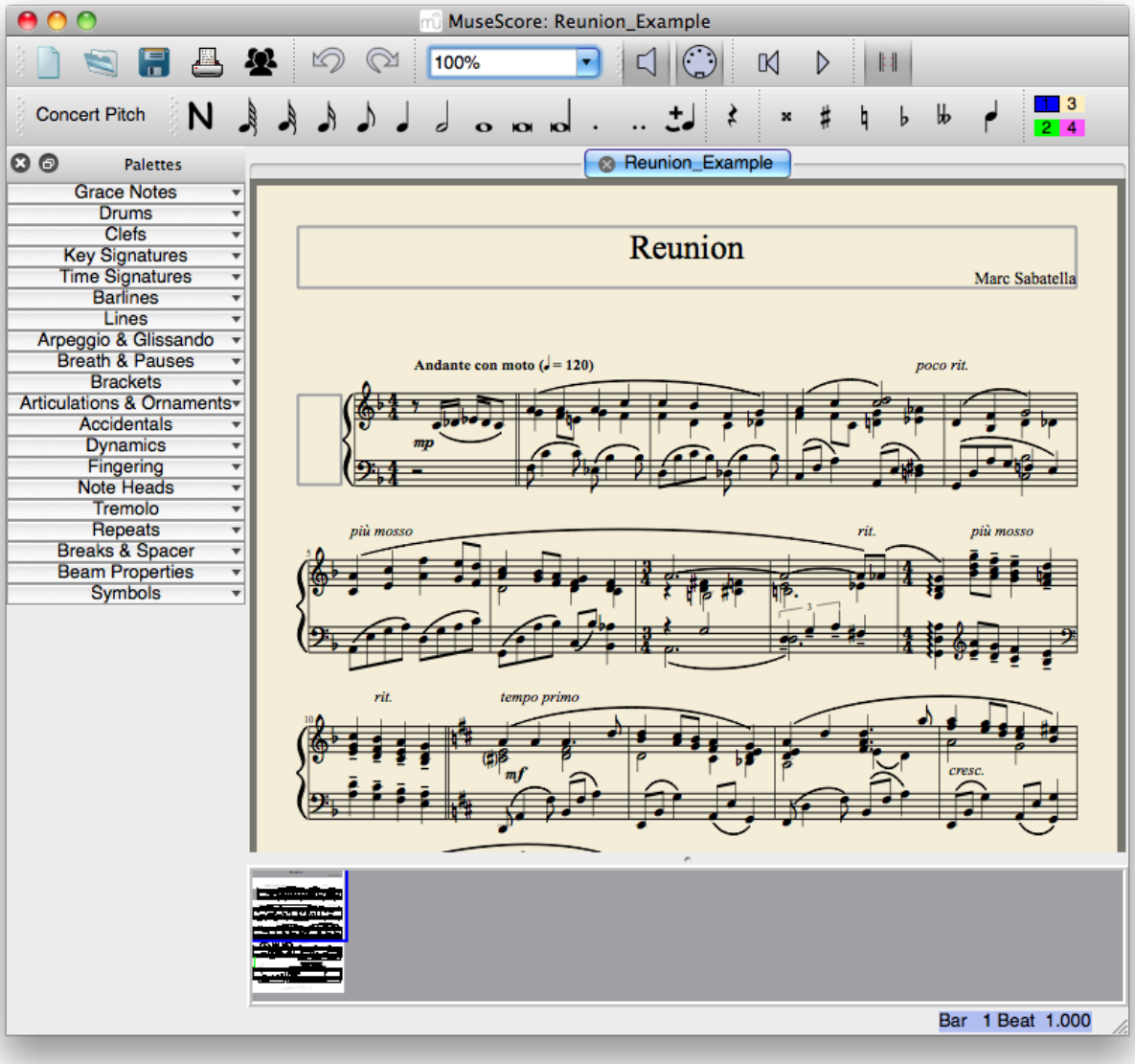

Figure 20: A screenshot of *MuseScore*. A kind of MS Word for music.

produces tones with roughly equal overall pitch height and with no clear pitch maxima or minima. In my experiments, I approximated this by using constant amplitude, $r_x = 0.1\,\frac{80}{127}$, within natural oscillator frequency range of the GFNN-ANN, and zero amplitude otherwise. To avoid the effect of limited range of natural oscillator frequencies (Bb2 – D5), each pair was transposed so that the corresponding attraction was directed towards C. Since simulating four seconds of real time with my current implementation of the GFNN-ANN is an overkill, I used stimuli which were only half a second long. These were produced and rendered as MIDI files with *MuseScore* [25] (see Figure 20), then streamed to my program with *MidiPipe* [32] (see Figure 21).

results

The stimuli together with corresponding snapshots[8] of the GFNN-ANN

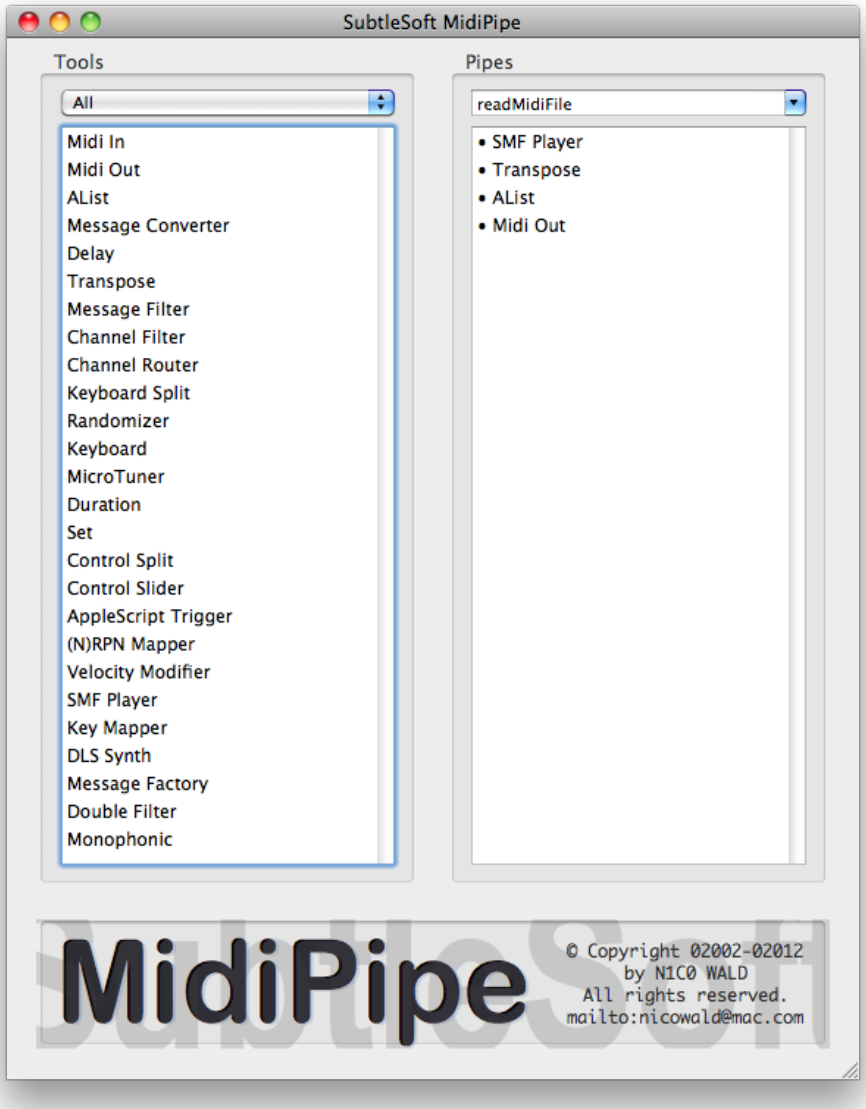

Figure 21: A screenshot of *MidiPipe*. All available primitive midipipes are listed on the left. Those which make up the midipipe used for streaming MIDI files to my app are listed on the right. MIDI flows from SMF Player via Transpose and AList to Midi Out. First, SMF Player converts a list of MIDI files to stream of MIDI messages. Next, Transpose transposes all pitches in the stream by the specified number of semitones (set to 0 during the experiments). Then, AList displays all MIDI messages in the stream on the screen. Finally, Midi Out sends the stream to a MIDI output device (for instance, a virtual MIDI bus polled for MIDI messages by my app).

are displayed in Figure 22, Figure 23, Figure 24, Figure 25, Figure 26, and Figure 27. All the constituent sinusoids are notated, so that even single pitches are notated as chords and will be refered to as such. Within each pair, the accented chord is the one which attracts the other in the pair more strongly. Each snapshot was taken approximately at the middle of the corresponding beat and represents values averaged over 0.1s. Within each snapshot, the first layer of the GFNN-ANN is rendered on the lower and the second on the upper grand staff. Natural oscillator frequency and instantaneous oscillator frequency were both transformed to MIDI note number[9]. The former is mapped onto horizontal and the latter onto vertical position on the grand staff. The gaps are caused by nonlinearity of traditional mapping from pitch to staff position. Instantaneous amplitude is mapped onto opacity. Note that the third stimulus was not actually used in the original study. I fabricated it to test behavior of the network outside the twelve-tone equal temperament. However, since, musically speaking, it is only a slight 'flattening' of the second one, we may argue that it will be perceived similarly. Overall, we can see that during stimulation, the salient pitches in both layers of the GFNN-ANN reflect those in the stimulus. The pitches are stable over the beat so that the snapshot could be taken anywhere within the beat and remain virtually the same. The pitches the second layer of the GFNN-ANN is attracted to when the stimulus has ceased and the first layer is silent always form a proper subset of those in the accented chord. Except C3, C5, and, to a lesser extent, C4, they are quite unstable so that the exact timing of the snapshot makes a difference.

testing GFNN-ANN for modulation

For the last experiment, I concocted a modulating chord progression (see Figure 28). If the network were to follow the rules from standard harmony textbooks, its second layer would have to be attracted to the first chord just after the second, and to the fourth just after the fifth.

results

Similarly to the previous experiments, during stimulation, the salient pitches in both layers of the GFNN-ANN reflect those in the stimulus. The pitches are stable over the beat so that the exact timing of the snapshot does not make any difference. Unfortunately, both attractions are directed towards G2 and G5 (see Figure 28). You may have noticed that for this experiment, I extended the range of natural oscillator frequencies (to F2 – A5). This was because with the original range both attractions were directed towards C3 and C5. A not very encouraging hypothesis emerges, discussed in the following paragraph.

discussion

The results of the first three experiments would suggest that the GFNN-ANN indeed can account for attractions of various scale degrees to tonic and does work even for tones (chords) outside the twelve-

8 These are actual screenshots of the app.

9 More precisely, the continuous extension of it.

tone equal temperament. However, in addition to the failure to simulate modulation, the last experiment indicates that the exact range of natural oscillator frequencies of the GFNN-ANN is a stronger determinant of its attractors then the stimuli. This hypothesis casts doubt on the results of the first three experiments. More specifically, the behavior of the GFNN-ANN probably wouldn't generalize to attractions directed towards other chords than C major.

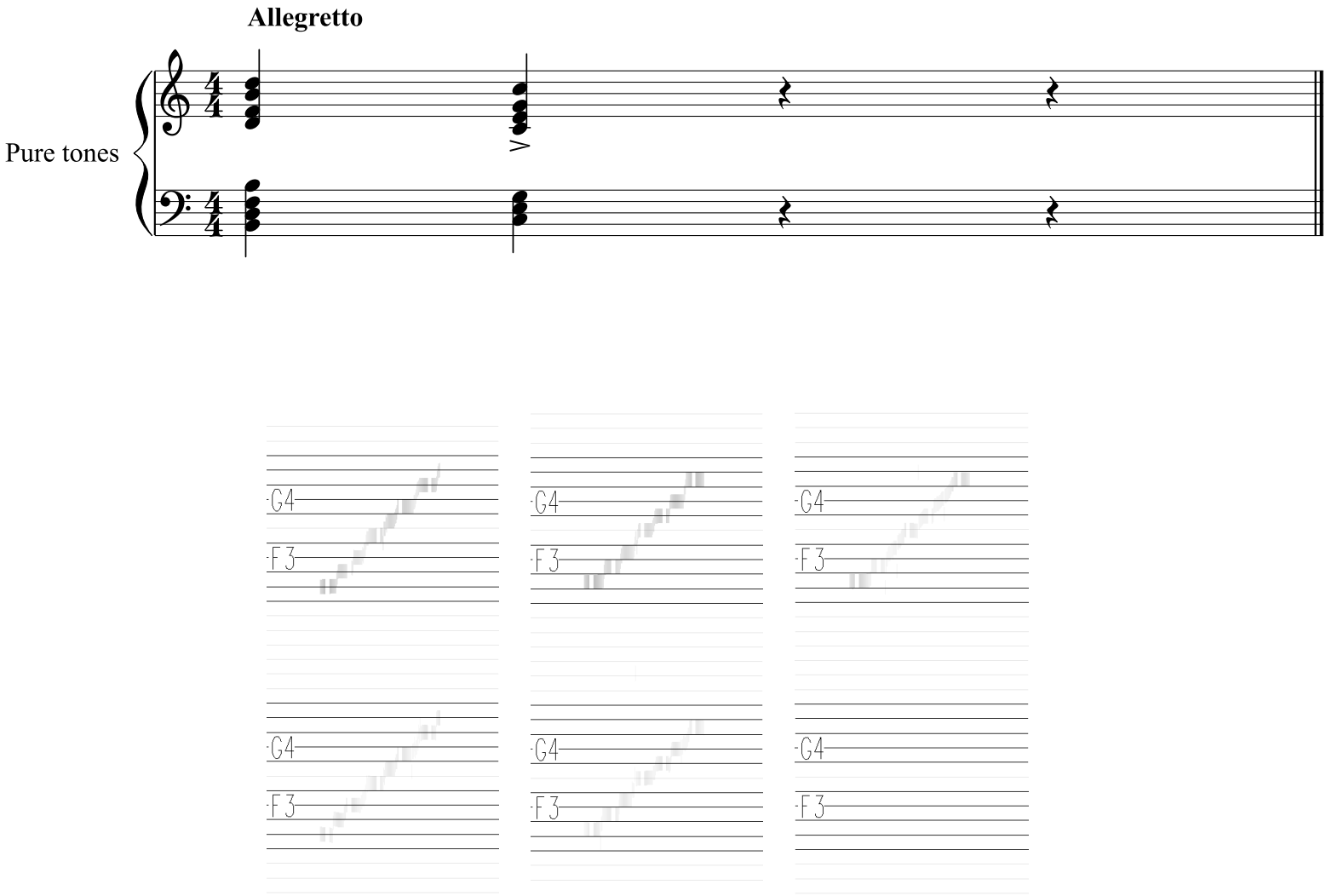

Figure 22: Experiment 1, temporal order 1. Top: the stimulus. Bottom: snapshots of the GFNN-ANN. During stimulation, the salient pitches in both layers reflect those in the stimulus. The pitches the second layer is attracted to when the stimulus has ceased and the first layer is silent form a proper subset of those in the accented chord.

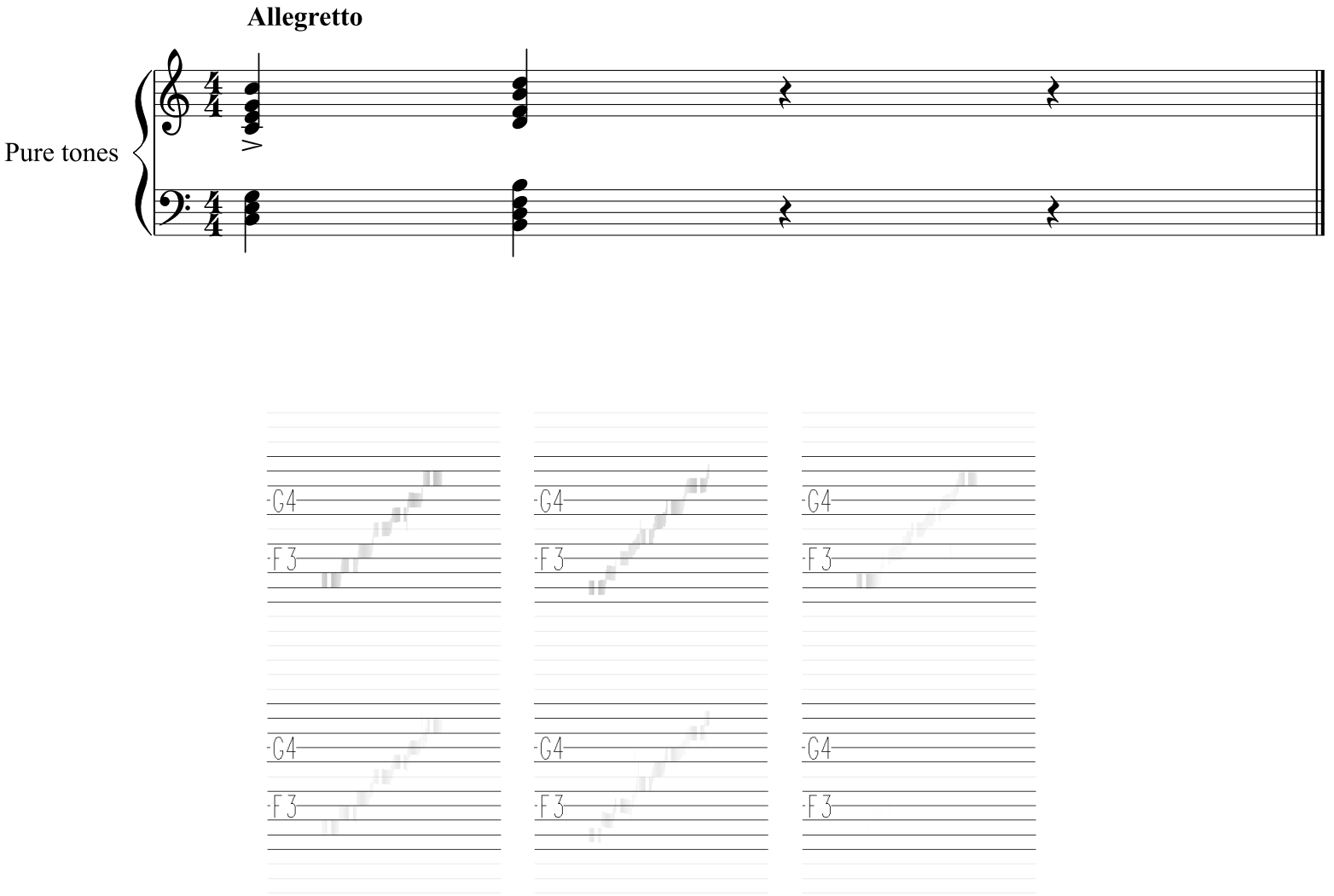

Figure 23: Experiment 1, temporal order 2.Top: the stimulus. Bottom: snapshots of the GFNN-ANN. During stimulation, the salient pitches in both layers reflect those in the stimulus. The pitches the second layer is attracted to when the stimulus has ceased and the first layer is silent form a proper subset of those in the accented chord.

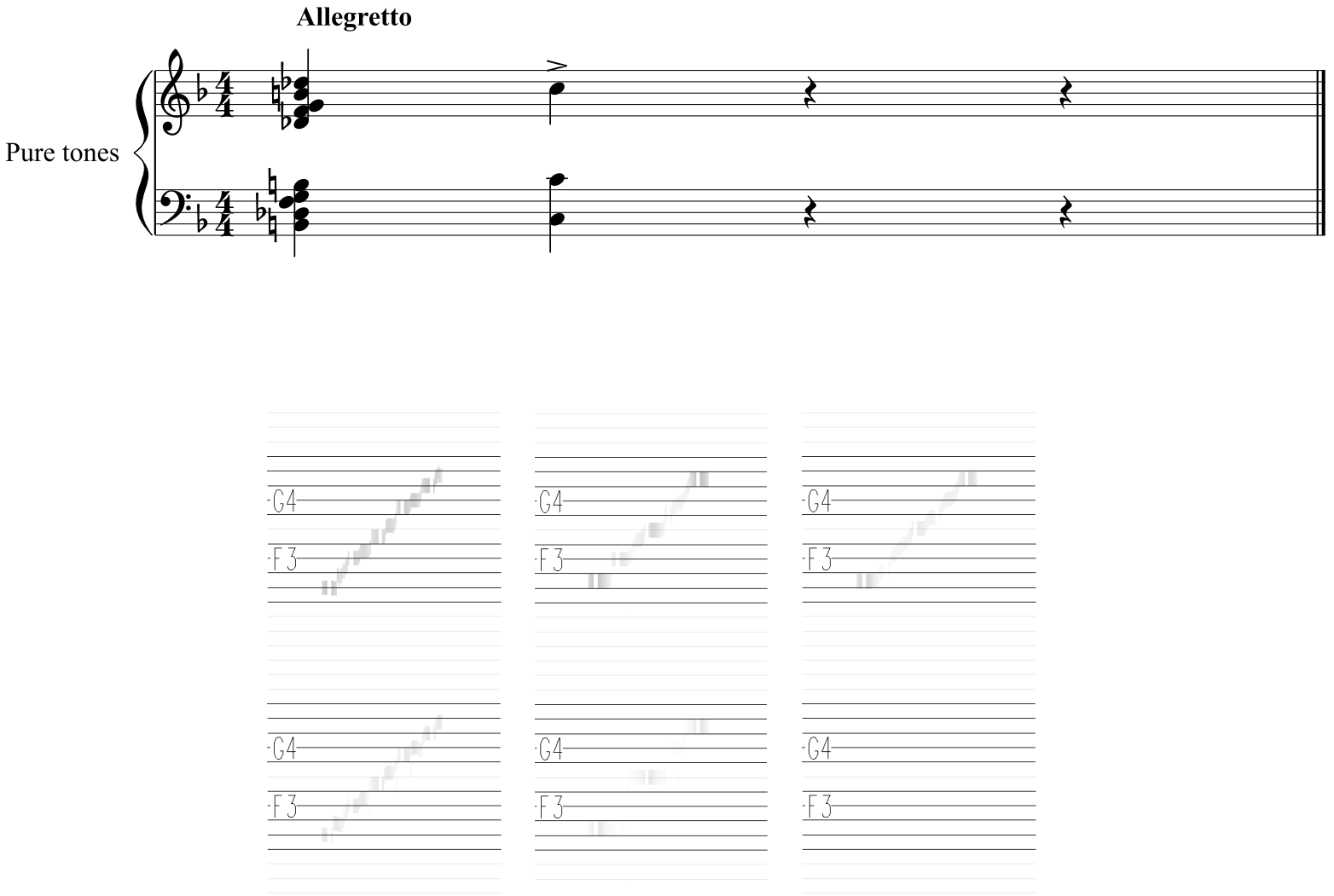

Figure 24: Experiment 2, temporal order 1.Top: the stimulus. Bottom: snapshots of the GFNN-ANN. During stimulation, the salient pitches in both layers reflect those in the stimulus. The pitches the second layer is attracted to when the stimulus has ceased and the first layer is silent form a proper subset of those in the accented chord.

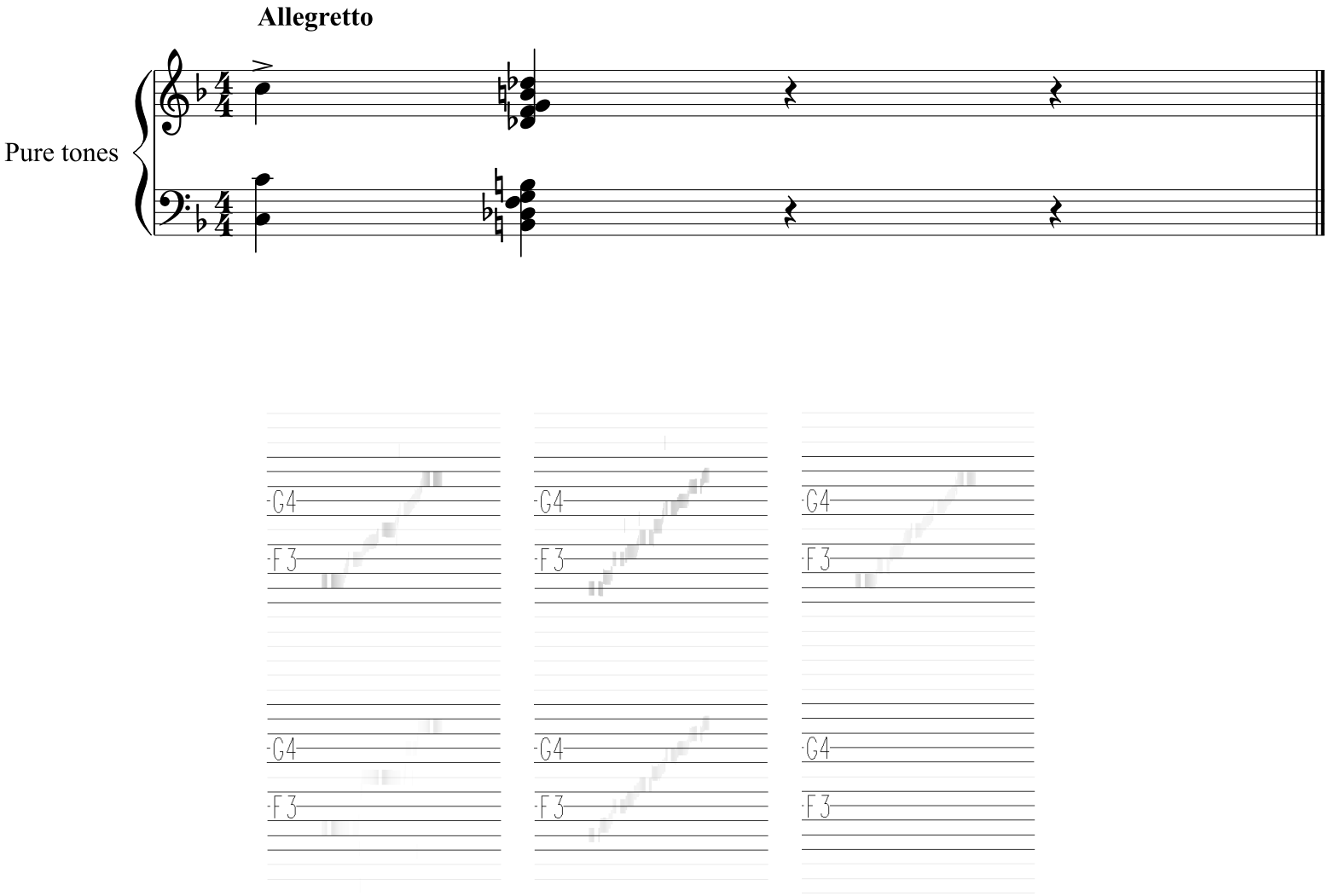

Figure 25: Experiment 2, temporal order 2.Top: the stimulus. Bottom: snapshots of the GFNN-ANN. During stimulation, the salient pitches in both layers reflect those in the stimulus. The pitches the second layer is attracted to when the stimulus has ceased and the first layer is silent form a proper subset of those in the accented chord.

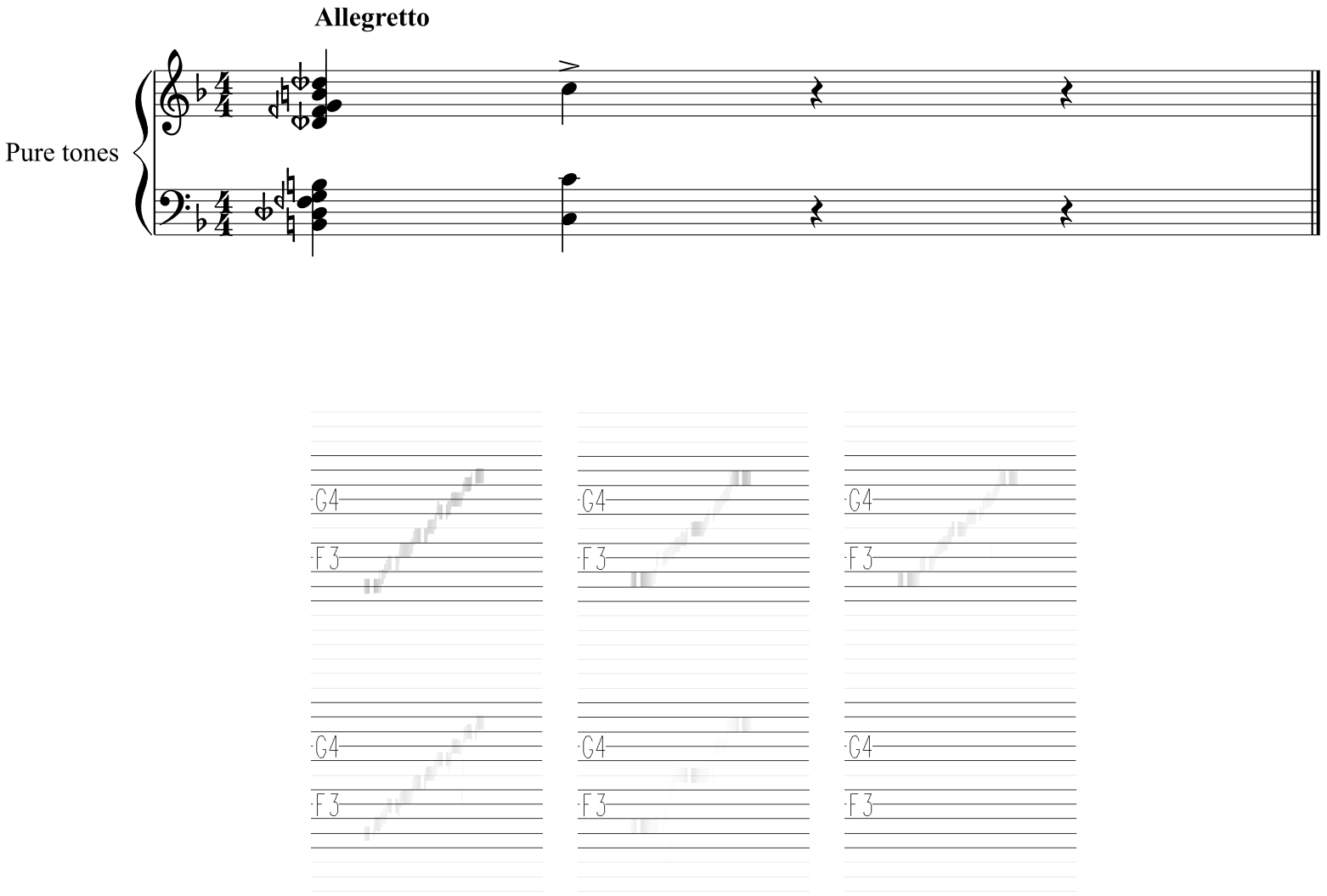

Figure 26: Experiment 3, temporal order 1.Top: the stimulus. Bottom: snapshots of the GFNN-ANN. During stimulation, the salient pitches in both layers reflect those in the stimulus. The pitches the second layer is attracted to when the stimulus has ceased and the first layer is silent form a proper subset of those in the accented chord.

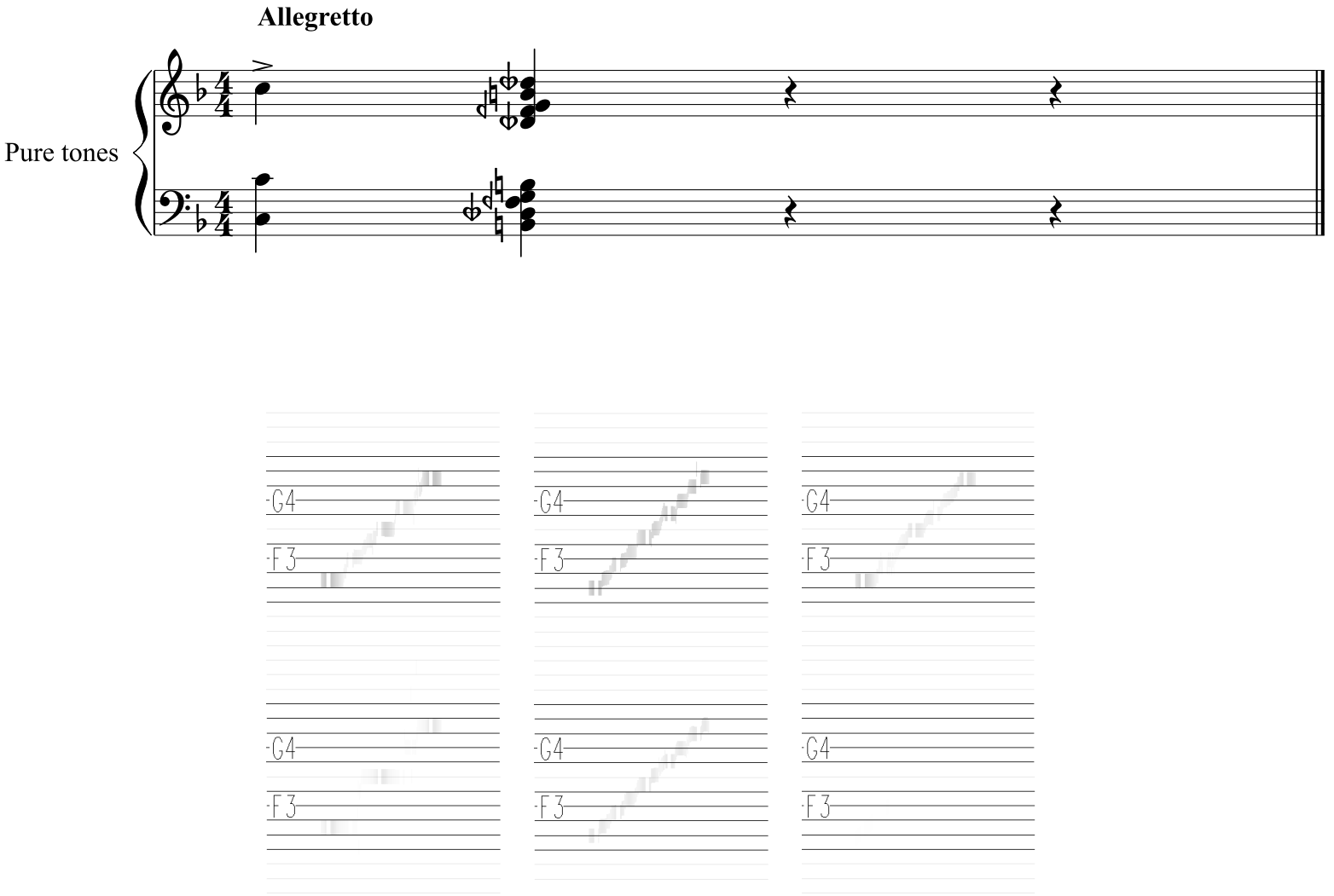

Figure 27: Experiment 3, temporal order 2.Top: the stimulus. Bottom: snapshots of the GFNN-ANN. During stimulation, the salient pitches in both layers reflect those in the stimulus. The pitches the second layer is attracted to when the stimulus has ceased and the first layer is silent form a proper subset of those in the accented chord.

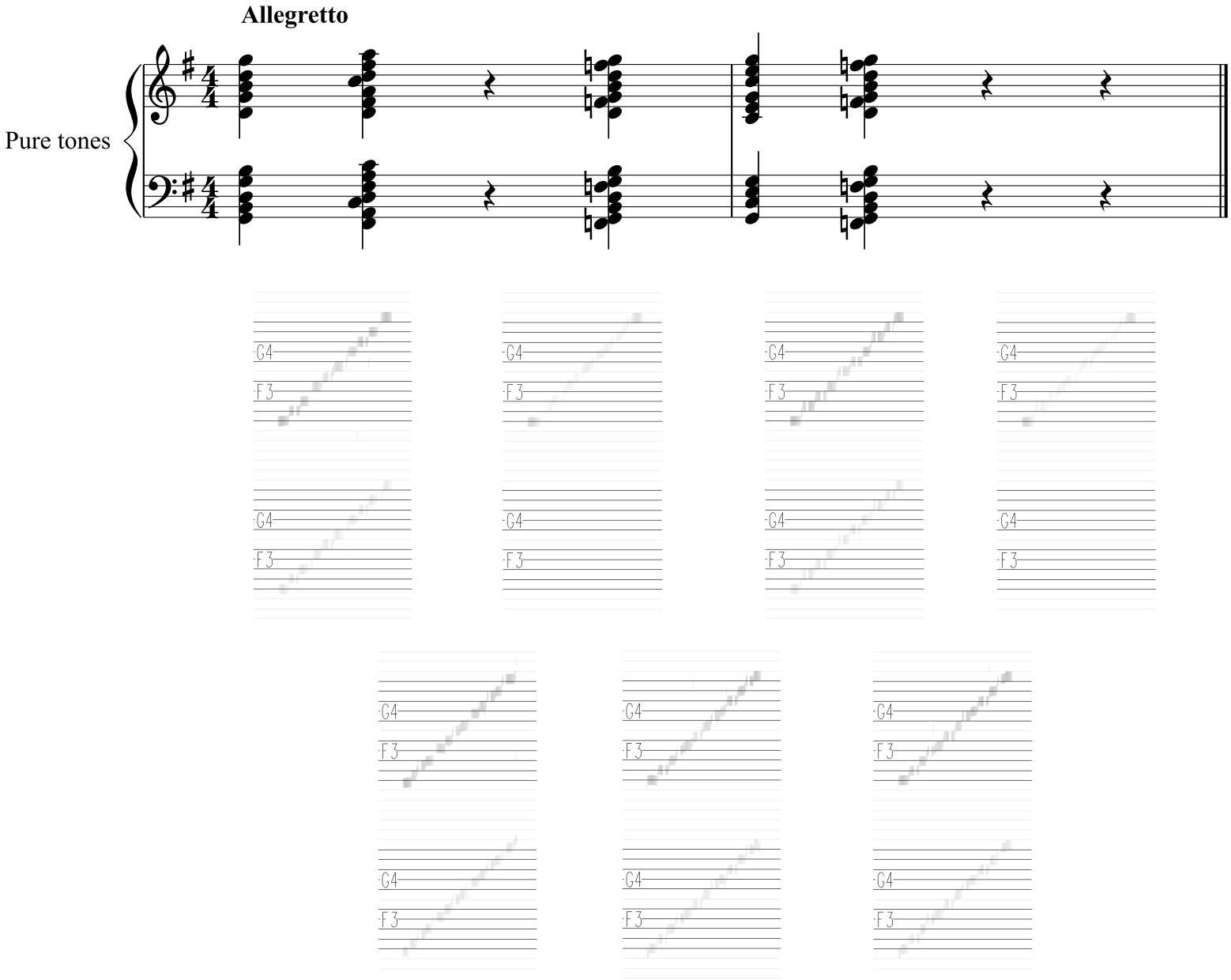

Figure 28: Experiment 4.Top: the stimulus. Bottom: snapshots of the GFNN-ANN. The GFNN-ANN does not reflect the change of key.

There are very few things which we know,
which are not capable of being reduc'd to a Mathematical Reasoning;
and when they cannot
it's a sign our knowledge of them is very small and confus'd [...]

— John Arbuthnot [2]

5 CONCLUSION

how I have familiarized myself with GFNN-ANNs

According to the assignment, my first task was to familiarize myself with the subject of gradient-frequency neural networks (GFNNs). My study of gradient-frequency-neural-network-based artificial neural networks (GFNN-ANNs) in general is summarized in Chapter 2. More precisely, GFNN-ANN 'neuron' is explained briefly, as is GFNN-ANN structure. With respect to the rest of the assigment, in my study of particular GFNN-ANNs, I have focused on those suitable for modeling tonal attractions. Therefore, the second gradient-frequency-neural-network-based artificial neural network from [16] (Network 2), the only GFNN-ANN I know to be capable of modeling a tonal attraction, is discussed in Chapter 2. More specifically, the equations for its 'neuron' are reproduced, their derivation summarized and their parameters explained. Additionally, Network 2 structure is described. A GFNN-ANN capable of modeling beat perception is touched upon in Chapter 3.

how I have tested GFNN-ANN for tonal attractions

My second and last task was, as stated in the assignment[1], to make experiments in an attempt to answer especially the following questions:

1. Which of the tonal relationships (read tonal attractions) perceived by humans while listening to music can be modeled using GFNN (read GFNN-ANN)? Is it confined to the relationship between subtonic (read leading tone) and tonic in the major scale or does it encompass, e.g., chromatic scale degrees?

2. When stimulated with a modulating musical phrase, does GFNN (read GFNN-ANN) reflect the change of key?

I had the choice to test the model against either existing empirical data from experiments (with human subjects) or rules from standard harmony textbooks, as convenient. The first question was addressed by experiments 1 and 2 from Chapter 4. Experiment 1 tested a GFNN-ANN for attraction from a diatonic chord[2] to tonic and experiment 2 tested the GFNN-ANN for attraction from a chromatic chord[3]

1 In the following restatement, some terms are clarified/corrected.
2 That is, one that does not contain any chromatic scale degrees.
3 That is, one that does contain some chromatic scale degrees.

to tonic. For both experiments, the attractions were taken from an experiment with human subjects. Both attractions have indeed been modeled succesfuly with the GFNN-ANN. However, generalization to attractions directed towards other chords than C major is problematic. The second question was adressed by experiment 4 from Chapter 4. It tests the GFNN-ANN for modulation. This time, the modulation reflected standard rules of harmony. The GFNN-ANN didn't reflect the change of key.

how I have put GFNN-ANNs in context of contemporary music composition

Beyond the assignment, I have put GFNN-ANNs in the context of contemporary music composition. Particularly, in Chapter 1, I have sketched the problem of reconciliating innovation with communication and acknowledged the potential of GFNN-ANNs to simulate attractions pertaining to any musical sound and thus provide contemporary composers with a code for communicating emotions through novel sounds. Consequently, Chapter 3 gives an account of my attempt to implement a GFNN-ANN as a utility for contemporary composers, that would enable them to test their ideas against this code interactively. While striving for a solid performance, I got my hands dirty with such peculiarities as the actor model of concurrent computation, CUDA, signal functions or software transactional memory. The final implementation from Chapter 3 was then used in the experiments from Chapter 4. Also, experiment 3 from Chapter 4 tested the GFNN-ANN for attraction from a chord outside the twelve-tone equal temperament to tonic. The attraction indeed took place, but the problem of generalization applies here, too.

what to do next

What to do next? With the final implementation from Chapter 3, even several months of parameter tuning were not enough to bring a GFNN-ANN to fulfill the potential satisfactorily, as testified by Chapter 4. I qualify the statement with the implementation rather than with the model, since I strongly believe there must a way to make the GFNN-ANNs work. After all, our brain is not much more than a network of nonlinear oscillators (neurons). Consequently, the parameter space of GFNN-ANNs is, in my opinion, worth exploring further. The implementation will be a bottleneck of any black box method of parameter optimization. Even if we tackle the problem analytically, the app will still be too slow to be useful to composers. Therefore, my future efforts will be directed towards making the app run faster. How can it be done? One could further tune the (mostly) sequential implementation. However, there does not seem to be much room for improvement in this direction, since even now, e.g., the app virtually does not allocate at all after initialization. Alternatively, massive parallelization could be employed. However, for the time being, it is rather hard. Finally, a mathematical approach could come to the rescue, like the one exemplified by Wilson and Cowan [33]. They modeled the dynamics of cortical tissue containing hundreds of thousands of neu-

rons with a system of two differential equations. It would be wonderful to be able to model the GFNN-ANN dynamics so succinctly.

BIBLIOGRAPHY

[1] G.A. Agha. Actors: a model of concurrent computation in distributed systems. 1985. (Cited on page 19.)

[2] J. Arbuthnot, J. Ham, and C. Huygens. *Of the laws of Chance, or, a Method of Calculation of the Hazards of Game*. Motte and C. Bathurst, 1738. (Cited on page 41.)

[3] MIDI Manufacturers Association. An Introduction to MIDI. http://www.midi.org/aboutmidi/intromidi.pdf. (Cited on page 3.)

[4] M.M.T. Chakravarty, G. Keller, S. Lee, B. Lever, T.L. McDonell, R. Newtown, and S. Seefried. The accelerate package. http://hackage.haskell.org/package/accelerate, . (Cited on page 19.)

[5] M.M.T. Chakravarty, G. Keller, S. Lee, and T.L. McDonell. The accelerate-cuda package. http://hackage.haskell.org/package/accelerate-cuda, . (Cited on page 19.)

[6] M.M.T. Chakravarty, G. Keller, S. Lee, T.L. McDonell, and V. Grover. Accelerating Haskell array codes with multicore GPUs. In *Proceedings of the sixth workshop on Declarative aspects of multicore programming*, pages 3–14. ACM, 2011. (Cited on page 20.)

[7] J. Dagit and S. Panne. The OpenGL package. http://hackage.haskell.org/package/OpenGL-2.6.0.1. (Cited on page 22.)

[8] H.W.C. Davis, editor. *Aristotle's Politics*. Clarendon press, 1926. (Cited on page 8.)

[9] G.B. Ermentrout, C.C. Chow, et al. Modeling neural oscillations. *Physiology and Behavior*, 77(4):629–634, 2002. (Cited on page 8.)

[10] T. Harris, S. Marlow, S. Peyton-Jones, and M. Herlihy. Composable memory transactions. In *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 48–60. ACM, 2005. (Cited on page 21.)

[11] HaskellWiki. Numeric Haskell: A Vector Tutorial. http://www.haskell.org/haskellwiki/Numeric_Haskell:_A_Vector_Tutorial. (Cited on page 20.)

[12] P. Hudak. Euterpea. http://cs.yale.edu/homes/euterpea/Euterpea/. (Cited on page 20.)

[13] D. Huron. *Sweet anticipation: Music and the psychology of expectation*. MIT press, 2006. (Cited on page 1.)

[14] S.P. Jones, A. Gordon, and S. Finne. Concurrent Haskell. http://hackage.haskell.org/packages/archive/base/4.5.0.0/doc/html/Control-Concurrent.html. (Cited on page 21.)

[15] Y. Kuramoto. In *International Symposium on Mathematical Problems in Theoretical Physics*, Lecture Notes in Physics, Volume 39, page 420. Springer, Berlin, 1975. (Cited on page 8.)

[16] E.W. Large. A dynamical systems approach to musical tonality. *Nonlinear Dynamics in Human Behavior*, pages 193–211, 2011. (Cited on pages xiii, 1, 14, and 41.)

[17] E.W. Large. Musical tonality, neural resonance and Hebbian learning. *Mathematics and Computation in Music*, pages 115–125, 2011. (Cited on page 14.)

[18] E.W. Large, F.V. Almonte, and M.J. Velasco. A canonical model for gradient frequency neural networks. *Physica D: Nonlinear Phenomena*, 239(12):905–911, 2010. (Cited on pages 8 and 10.)

[19] R. Leshchinskiy. The vector package. http://hackage.haskell.org/package/vector-0.9.1. (Cited on page 20.)

[20] M. Lipovaca. *Learn You a Haskell for Great Good!: A Beginner's Guide*. No Starch Press, 2011. (Cited on page 18.)

[21] P.H. Liu. The PortMidi package. http://hackage.haskell.org/package/PortMidi. (Cited on page 21.)

[22] P.H. Liu and M. Sunet. The GLFW package. http://hackage.haskell.org/package/GLFW-0.5.1.0. (Cited on page 22.)

[23] S. McClary. *Feminine endings: Music, gender, and sexuality*. University of Minnesota Press, 2002. (Cited on page 28.)

[24] T. Murail. Winter Fragments. http://www.tristanmurail.com/en/oeuvre-fiche.php?cotage=27488. (Cited on page 1.)

[25] W. Schweer et al. MuseScore. http://musescore.org/en. (Cited on page 30.)

[26] D. Smith. A Haskell lover's plea. http://www.haskell.org/haskellwiki/Humor/Poem. (Cited on page 18.)

[27] N. Steinbeis, S. Koelsch, and J.A. Sloboda. The role of harmonic expectancy violations in musical emotions: Evidence from subjective, physiological, and neural responses. *Journal of Cognitive Neuroscience*, 18(8):1380–1393, 2006. (Cited on page 1.)

[28] M. Sulzmann. The actor package. http://hackage.haskell.org/package/actor. (Cited on page 19.)

[29] M. Sulzmann, E. Lam, and P. Van Weert. Actors with multi-headed message receive patterns. In *Coordination Models and Languages*, pages 315–330. Springer, 2008. (Cited on page 19.)

[30] W.N. Thornton and T. DuBuisson. The stm-chans package. http://hackage.haskell.org/package/stm-chans. (Cited on page 21.)

[31] M.J. Velasco and E.W. Large. Pulse detection in syncopated rhythms using neural oscillators. *pulse*, 1(2):3–4, 2011. (Cited on page 18.)

[32] N. Wald. MidiPipe. http://www.subtlesoft.square7.net/MidiPipe.html. (Cited on page 30.)

[33] H.R. Wilson and J.D. Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, 12(1):1–24, 1972. (Cited on pages 8 and 42.)

[34] M.H. Woolhouse and I. Cross. An interval cycle-based model of pitch attraction. In *Proceedings of the 9th International Conference on Music Perception & Cognition*, pages 763–771, 2006. (Cited on pages 1 and 29.)

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured at:

http://postcards.miede.de/

Final Version as of January 9, 2013 at 1:02.

DECLARATION

I hereby declare that I prepared this master thesis independently and on my own, by exclusive reliance on the information sources indicated therein in accordance with *Metodický pokyn č. 1/2009: O dodržování etických principů při přípravě vysokoškolských závěrečných prací*.

Prague, January 3, 2013

Michal Hadrava

PROHLÁŠENÍ

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s *Metodickým pokynem č. 1/2009: O dodržování etických principů při přípravě vysokoškolských závěrečných prací*.

Praha, 3. ledna 2013

Michal Hadrava

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 3. 1. 2013 .....

.....

Podpis autora práce