PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO · COLEÇÃO PESQUISA UFES ·

VOLUME 38

LEANDRO COSTALONGA

Biomechanical modelling of guitar performance

EDUFES

Esta obra foi selecionada para integrar a "Coleção Pesquisa Ufes", a partir de Chamada Pública feita pela Pró-Reitoria de Pesquisa e Pós-Graduação (PRPPG) da Universidade Federal do Espírito Santo (Ufes) aos programas de pós-graduação da universidade.

A seleção teve por base pareceres que consideraram critérios de inovação, relevância e impacto.

O financiamento da Coleção foi viabilizado por meio do Programa de Apoio à Pós-Graduação (Proap) da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes) e de recursos do Tesouro Nacional.

EDUFES
EDITORA

Editora Universitária – Edufes

Filiada à Associação Brasileira
das Editoras Universitárias (Abeu)

Av. Fernando Ferrari, 514
Campus de Goiabeiras
Vitória – ES · Brasil
CEP 29075-910

+55 (27) 4009-7852
edufes@ufes.br
www.edufes.ufes.br

Diretor da Graúna Digital
Thiago Moulin

Supervisão
Laura Bombonato

Seção de edição e revisão de textos
Carla Mello | Natália Mendes
Manuella Marquetti | José Ramos | Stephanie Lima

Seção de design
Carla Mello | Bruno Ferreira Nascimento

Projeto gráfico
Edufes

Diagramação e capa
Bruno Ferreira Nascimento

Revisão de texto
MC&G Editorial

Fotografia da capa por
Jeremiah Higgins em
https://unsplash.com/.

Esta obra foi composta com
a família tipográfica Crimson Text.

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

C837b

Costalonga, Leandro.
Biomechanical modelling of guitar performance [recursos eletrônicos] / Leandro Costalonga. - Dados eletrônicos - Vitória, ES : EDUFES, 2023.
305 p. : il. ; 21 cm. - (Coleção Pesquisa Ufes ; 38)

Inclui bibliografia.
ISBN: 978-85-7772-528-1
Modo de acesso: https://repositorio.ufes.br/handle/10/774

1. Música e tecnologia. 2. Biomecânica. 3. Violão. I. Costalonga, Leandro. II. Título. III. Série.

CDU:789.9

Elaborado por Ana Paula de Souza Rubim – CRB-6 ES-000998/O

LEANDRO COSTALONGA

Biomechanical modelling of guitar performance

EDUFES

Vitória, 2023

This book was selected to be part of the PRPPG books 'collections
"Coleção Pesquisa Ufes", funded by the PROAP program with
resources from the Brazilian national treasure.

Acknowledgments

To my wife, my parents, my children, and my friends. A loving thanks for all the support you give me. Also, a special thanks to all fellow researchers of ICCMR, G-ubimus and NESCoM.

This book originates from a PhD thesis entitled "Biomechanical Modelling of Musical Performance: A Case Study of the Guitar (2009)" written by the same author and supersized by Prof. Eduardo Reck Miranda and Dr. John Matthias at the University of Plymouth (UK). The original research was funded by the Brazilian Government's "Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior" (CAPES).

About the Author

Leandro Costalonga has a Computer Science Degree with Masters (UFRGS/Brazil) and PhD (University of Plymouth/UK) in Computer Music. Associate professor at the Federal University of Espírito Santo (UFES/Brazil) where teaches on undergraduate programs in Computer Science and Computer Engineering and Graduate Program in Arts. Head of the NESCoM Research Group that carries on Computer Music related research, especially on Ubiquitous Music. Besides Computer Music, other research interest includes Human-Computer Interaction, Programming Languages and Artificial Intelligence

Preface

As early as the 1840s, mathematician and allegedly the first ever software programmer, Lady Ada Lovelace, predicted in that machines would be able to compose music. On a note about Charles Babbage's Analytical Engine, she wrote:

"Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the Engine might compose elaborate and scientific pieces of music of any degree of complexity or extent." (Manabrea 1843, p.21)

People hardly ever realize that musicians started experimenting with computing far before the emergence of the vast majority of scientific, industrial and commercial computing applications in existence today. For instance, in the 1940s, researchers at Australia's Council for Scientific and Industrial Research (CSIR) installed a loudspeaker on their Mk1 computer to track the progress of a program using sound. Subsequently, Geoff Hill, a mathematician with a musical background, programmed this machine to playback a tune in 1951.

That is not yet programming a computer to compose algorithmically. It is rather coding a piece of music for playback, as if the machine was the equivalent of one of those mechanical pianolas often spotted in saloons of Western films. Nevertheless, I would say this is when the field of computer music began.

For the last 70 years or so the field flourished at rapid speed. Today, it is impossible to think of any aspect of music making, distribution and listening that would not involve some form of computing technology or Artificial Intelligence (AI). And like it or not, computers furnished with AI can compose music automatically nowadays.

The music that those AI systems produce are very good. They may not always fulfil everyone's expectations and tastes, but machines have been capable of producing music for film, advertising and

even classical orchestral music that conned audiences to think they were composed by a human being.

One aspect of music that computers cannot still imitate humans well is performance. An artificially composed piece of music for, say, the guitar, sounds good when performed by a human guitarist. Give it to a computer to play and one can almost immediately tell it is artificial.

Programming a machine to imitate the way humans perform music is very difficult. Research in this area is relatively incipient. In this book, Leandro presents an approach to for doing this, which is a promising way forward.

The book presents a study on capturing, analyzing and modelling information about motor and biomechanical processes of guitar performance. Leandro believes that actions originated from the motor and biomechanical functions during a musical performance can provide important information for training an Artificial Intelligence system to perform music as expressively as a human being would do.

The bulk of this work emerged from the research which Leandro developed for his doctorate at the Interdisciplinary Centre for Computer Music Research of University of Plymouth, in the United Kingdom. I was most honoured to be his thesis supervisor. And I am very proud to see his research published as a book, which will certainly constitute an important reference for those interested in taking forward the fiendish task of modelling expressive musical performance.

Prof Eduardo Miranda
15 December 2020

Manabrea, L. F. (1843). Sketch of the Analytical Engine invented by Charles Babbage. Translated by Ada Lovelace. London, UK: R. & J. E. Taylor. Available online: https://johnrhudson.me.uk/computing/Menabrea_Sketch.pdf. Accessed on 03 Apr 2020.

Contents

Chapter 3

Chapter 4

Chapter 1

Introduction

> The world always makes the assumption that the exposure of an error is identical with the discovery of truth - that the error and truth are simply opposite. They are nothing of the sort. What the world turns to, when it is cured on one error, is usually simply another error, and maybe one worse than the first one. (Henry Louis "H. L." Mencken, 1949)

As early as the 1950s musicians started to gain access to computers to generate music. Pioneers include composers such as Lejaren Hiller, Gottfried Michael Koenig, Iannis Xenakis and Pietro Grossi, amongst others. The term algorithmic composition has become a synonym for music composed by a computer.

Composers working with algorithmic composition soon realised that performance information (for example, speeding up and slowing down while playing notes, and changing how loudly they are played) is very important in the creation of music by computers. Indeed, the first attempt at a computer music programming language, Music I, developed by Max Mathews at Bell Telephone Laboratories in 1957, was prompted by Mathew's wish to "write a program to

perform music on the computer" (Park, 2009, p.10). Apparently, this development began after Mathews and his boss, John Pierce, went to a piano concert together. One of the pieces was so badly performed that during the intermission Pierce suggested that perhaps a computer could do better. And Mathews accepted the challenge, a challenge which remains pertinent in computer music research today.

On the 22nd of November 2003, the 10th episode of the seventh and final season of Star Trek, a famous American science-fiction entertainment series written by Eugene Wesley "Gene" Roddenberry in 1966 and adapted to television by Dan Koeppel, was broadcast. In this episode, an android known as Lt. Cmdr. Data (played by Brent Spiner) was playing the Handel's Suite #7 in G Minor on the violin to "his" creator: Dr. Tainer (played by Fionnula Flanagan). The following dialog took place after the performance finished:

> *Dr. Tainer delightedly laughs and cheers Data, clapping enthusiastically.*
> "Thank you", said Data, "I'll be playing this piece at a recital tomorrow evening"
> "That was beautiful", said Dr. Tainer with an emotional voice.
> Data replied with a surprised face, "Hmm…I've been told that my playing is technically flawless but no one ever described it as beautiful".
> Dr. Trainer reassures Data, "It was…really!"
> "Are you certain you not saying that because you are my mother?" Data asks still not entirely convinced.
> Dr. Tainer laughs while Data continues, "I have noticed that parents tend to exaggerate when it comes to their children's accomplishments."
> "Well…I suppose there is a certain amount of vanity involved considering that giving you a creative aspect was my idea", Dr. Tainer proudly says.
> Data seems confused but she rapidly clarifies, "Your father did not really see the point. He believed that, since you don't have

> emotions, there will be no use for you to really express yourself", she pauses, looks into Data's eyes, and continues "somehow I had the feeling the opposite would be true".
> Data nudges in agreement, declaring "I do not know for certain but I believe it is during my creative endeavours that I come closest to experiencing what it must be like to be human"
> The dialogue finishes with Dr. Tainer offering to perform a duet with Data on the recital of the following day and Data accepting. As a result, they start to rehearse.

The story continues with the duet presentation at the recital, which has left the audience astonished. Data, however, seemed to be suspicious about the way Dr. Tainer was performing the violin. Data's suspicion was only explained after Dr. Tainer was left unconscious by accident.

During the medical procedures that attempt to revive Dr. Tainer, the medical crew noticed that despite the vital signals appear to be absolutely normal, there was something not right. It was only then that Data suggested that Dr. Tainer was also an android, which was confirmed after her skull was opened.

Surprised, Cmdr. William T. Riker (played by Jonathan Frakes) questioned Data about how he knew she was an android. The answer Data gave resonates with most of the current research in the field of expressive music performance modelling. He said:

> [...] we've practised the piece and I've noticed she played the same way during the performance. Every pitch, every intonation was exactly the same. Only an artificial life form could have done that.

Performers do have an impressive ability to replicate the expressive profile of a piece in performance, with a degree of variability in the timing properties of a performance of one per cent or less (Clarke, 1993). However, as demonstrated by Palmer (1997), performers

cannot control all the variations in the performance, even when they try to play with no "expression" whatsoever.

Measuring the deviation of the musical performance from what is actually written on the score is the most common technique to quantify the "expressiveness" of the performance (Sundberg et al., 1983a; Sundberg et al., 1983b) but according to Parncutt (1997), expressive variation is more than just a deviation from, or a distortion of, the original (notated) piece of music.

The personal motivation behind this research ultimately contributes to the development of technology which is capable of artificially performing music as a human would, in special guitarists. Such a technology could be used by composers to predict how their compositions would sound when played by a particular guitarist without having to hire them. In an attempt to gain an initial practical understanding of the properties of a music performance produced by a human performer, we have developed a simple listening experiment aimed at assessing whether listeners could differentiate computer--generated from human played guitar performances.

Simple harmonic sequences of thirty-second's duration were produced by five distinct sources: three computer software (namely, Guitar Pro, Finale and Polvo), a professional guitarist, and an amateur guitarist. Even the samples produced by the human guitarist were generated by a digital sampler controlled by a MIDI guitar. That way, the listener would not be able to make any decision on the basis of acoustical properties. In total, thirty samples were produced, six per source.

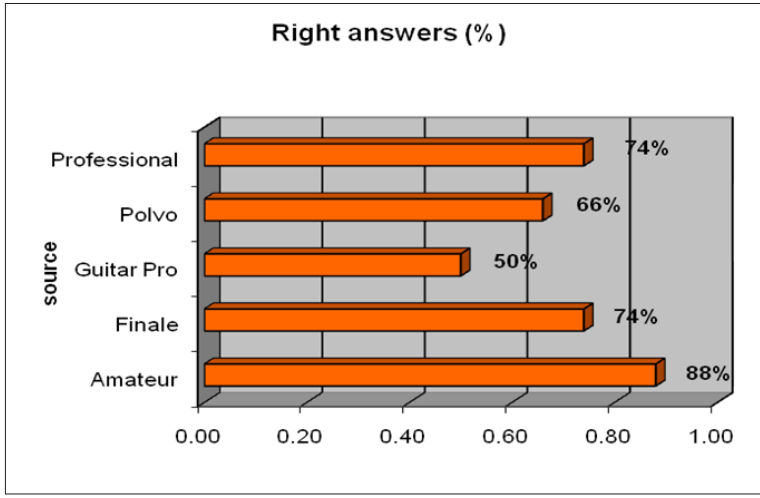
Twenty-five subjects took part in the experiment, fourteen of them self-professed musicians. A random choice of ten samples was presented to the subjects, two samples from each source.

The results have shown that 70.4% of the time listeners could determine whether a performance was generated by a computer or played by a human. Unlike Lt. Cmdr. Data, the android from Star Trek, the listeners did not have any previous information regarding

the way the sources produced the performance nor access to a musical score. So, how did they do it?

This question was put directly to the subjects, but instead of offering a precise answer they generally reported that their decisions were made based on "the way the notes flow" or "how familiar it sounds". This rather vague type of answer is not very helpful at first. However, after analysing the results an interesting point can be noticed: not only could the listener distinguish human from computer-generated performances, but they could also distinguish the amateur guitarist more easily, as shown in Figure 1.

Figure 1: Results of a preliminary listening experiment.



Source: Own image.

The x-coordinates show the percentage of the subject's correct classification per source (y-coordinates).

Figure 1 shows the percentage of the correct classification of the performer. Guitar Pro and Finale are commercial systems that required human expertise to generate the samples in a way that sounds natural. Polvo is a system of our own design, which is a precursor

of the Octopus Music API introduced in Chapter 5. Polvo does not require human intervention; it generates the samples autonomously. As we can see, the samples that subjects found harder to identify were programmed using a Guitar Pro, which is a software purposely built to formalise guitar performance. As previously said the samples that the subjects found easier to identify were generated by the amateur guitarist; they were positively identified by the listeners 88% of the time.

Compared with the performance recorded from the professional guitarist, the amateur produced a "dirtier" performance. The notes were overlapping each other, missing, or played out of time. Could the little imperfections commonly found in music performance contribute to the so-called "human-feel" in computer-generated music?

Poepel (2005) states that deviations communicate the performer's expressive intentions and emotions. However, the amount of deviations that are actually intentional is not clear. In fact, by just measuring the deviations it is not possible to distinguish between intentional and unintentional actions because they are the result of mixed cognitive and motor processes.

The cognitive aspects of musical performance have been widely studied by psychologists, musicologists, and even computer scientists, who have been using computer-modelling techniques to support cognitive theories. On the other hand, the motor and biomechanical aspects have not received much attention in such studies. In fact, only a few researchers actually study how the performer executes certain actions in an instrument with the intention of modelling them (Meister, 1989), therefore this will be our focus.

This book makes a contribution to the area of motor and biomechanical modelling of a musical performance focusing mainly on technical errors. However, we do acknowledge the importance of the cognitive models that focus on intentional variations during a musical performance with expressional purpose. Here, we are not stating that one approach should be preferred over the other; we are of the

opinion that biomechanical models (unintentional variations) are complementary to cognitive models (intentional variation). For instance, biomechanics can be seen as a "filter" that shapes the actions planned at the cognitive level. That is, the human body is viewed as some form of "bottleneck" based on the laws of (bio)physics which constrains the abstract and culturally specific principles of composition and performance (Clarke, 1993).

WHY SHOULD WE DISCUSS BIOMECHANICS IN GUITAR PERFORMANCE MODELLING?

The overarching questions that motivate our research are:

1. Do unintentional actions originating from the motor and biomechanical functions during music performance contribute a "human feel" to the performance?
2. Would it be possible to determine and quantify what such unintentional actions are?
3. Would it be possible to model and embed such information in computer systems for music performance?

The methodology to address these questions is comprised of the following:

1. Research into the mechanics, ergonomics, and playability of a given musical instrument. The focus is to understand the physical actions that must be performed in order to produce sounds, rather than focusing on the acoustical properties of the produced sound.
2. Once the actions (techniques) required for producing such sounds are understood, then the next step is to study how the human body conforms to these actions from a strictly motor and biomechanical viewpoint. The movements, postures, and the effort to perform are central to this investigation
3. Research into formalisation and modelling of motor and biomechanical performance information. The performance

actions can only be simulated by computer if they can be formalised in computational terms. This formalisation must include not only descriptors for musical actions but also cover mechanical aspects of the musical instrument and the human body when performing music.

The instrument chosen for this research is the guitar. There are good reasons for this choice: guitars are popular, mechanically simple, and inexpensive instruments that have been taught not only by the classical school but also in a less formal education, which led to different playing styles, techniques, and simplified musical notations that allow the guitarist to perform more freely (i.e. tablature). Despite these advantages, the guitar has seldom been chosen for research into computer models of music performance. Rather, instruments such as the piano, trumpet, sax, flute, violin and even drums have been preferred over the guitar (Bilitski, 2005; Dahl, 2006; Madsen and Widmer, 2006). By choosing the guitar we are certainly making a significant contribution to the field.

From the motor and biomechanical perspective, the guitar does not rank between the most studied human activities either. In the light of that, we decided to investigate, some of the attributes that we believe can influence the quality of a guitar performance are: force, speed, and precision of the digits of the left-hand (fretting hand). The biomechanical study of all aspects of the guitar performance, which would include the right-hand and the synchronization between the hands, is rather outside the scope of this book at this stage, given that there is almost no literature on the subject currently in existence.

Therefore, the objectives of this book are:

1. Gain a better understanding of the mechanics, ergonomics, and playability of the guitar, focusing on the physical actions that must be performed in order to play the guitar, rather than focusing on the acoustical properties of the sounds produced by this instrument.

2. Gain a better understanding of how the human body conforms to such physical performing actions (e.g. movements, postures, and the effort) from the motor and biomechanical viewpoint. In this research, we have focused on classical guitar techniques using the work of Abel Carlevaro (1984) as our main reference.
3. Develop a methodology to formalise quantifiable (i.e., acquire and analyse) data about the aforementioned physical performing actions on the guitar. In this book, the proposed methodology will be limited to the guitarist's left-hand.
4. Develop an approach to modelling the information formalized above, suitable for embedding such information in computer systems for music performance.
5. Demonstrate how the proposed modelling approach can effectively be embedded in computer systems for music performance.

HOW IS THIS BOOK ORGANIZED?

This book is dived into five main chapters, in addition to this introduction.

"Chapter 2 Approaches to Modelling Music Performance: A Literature Review" presents the background to the field of music performance modelling. It discusses expressive music performance and the two main approaches to modelling them: a) the simulation-based approach that just attempts to find patterns of deviations from the score; and b) the behavioural-based approach that can also be used to model music performance but focuses on the cognitive side. In Chapter 2, we also briefly discuss the role of errors in musical performance and how to produce computer-generated performances that "sound" realistic.

"Chapter 3 The Multiple Aspects of Guitar Performance" focuses on objectives 1 and 2. It explores the two main elements in a guitar

music performance: the guitar and the guitarist. The guitar is described in terms of its mechanical and playable properties; Like-wise, the performer is also analysed through a biomechanical and physiological view of the playing techniques taught by the classical guitar school. Other factors that could potentially interfere in musical performance are also mentioned.

"Chapter 4 Guitar Performance Data Acquisition and Analysis" focuses on objective 3. It describes the methodology used to capture and compile the data on a guitarist's left hand. Two experiments are reported: a) Speed and Precision; and b) Force and Posture. Since there is no commercial equipment suitable to measure force in guitar performance, we had to build our own device. Besides, a substantial amount of software development was necessary to process the data. Chapter 4 also reveals some interesting preliminary results that reinforce the idea that motor and biomechanical constraints do indeed play a role in the quality of musical performance.

"Chapter 5 Guitar Performance Modelling" focuses on objectives 4 and 5. This chapter is divided into two main sections. The first section addresses a Java Application Programming Interface (API), named Octopus Music API, which was designed to model music performance. The main classes of the API are presented with code examples to illustrate its use. In the second section, Machine Learning algorithms are discussed, proposed, and evaluated in the task of predicting the speed, force and precision of guitarists when performing guitar chords. There is still a third and fourth section that shows how the biomechanical-inspired models can be integrated with the Octopus Music API followed by an overall reflection over the solution proposed.

Finally, "Chapter 6 Conclusion" highlights the contribution to the knowledge introduced in the book and makes recommendations for future work.

Chapter 2

Approaches to Modelling Music Performance: A Literature Review

> I am speaking of things moved in the way that the voice is moved in speaking and singing, and the body in making a gesture and dancing… (Aristoxenus of Tarentum, elementa Rhythmica, c. 320 B.C;Todd, 1995a)

Musical performance provides a rich domain for the study of both cognitive and motor skills (Palmer, 1997). It is a means of communication involving three actors: the composer, the performer, and the listener (De Poli, 2004). The composer codifies musical ideas into a written notation (score); the performer transforms the score into an acoustic signal; and the listeners recode the acoustic signal back into ideas (Kendall and Carterette, 1990).

The vast majority of contemporary research on musical performance has focussed on perceptual processes of the listener since this is the focus of all musical activity (Sloboda, 2000); composition would

have no purpose if it were not experienced. The composer's part, the score, has long been studied and scrutinised in terms of its structural aspects such as harmony, melody, form, and instrumentation, as well as studying the composer's intention or the inherent emotional expression (Friberg *et al.*, 2006). However, the key to modelling music performance lies with the performer.

The performer interprets the symbolic information on the score and produces the sound by using a musical instrument (De Poli, 2004). The performing artist is an indispensable part of the system, shaping the music in creative ways by continually varying parameters like tempo, timing, dynamics (loudness), and articulation, in order to express their personal understanding of the music (Madsen and Widmer, 2006).

To model music performance we must first understand the processes the performer carries out to 'interpret' the piece of music. Only then will it be possible to artificially recreate this 'interpretational' behaviour. Two main approaches have been taken in attempting to do this:

The first approach simply searches for patterns of deviations between the input information the performer is given (the musical score) and the performance that is produced. The internal processes are not relevant as long as the behaviour can be simulated. We will refer to this approach as Simulation-Based Modelling.

The second approach is concerned with the internal processes of the performer. It attempts to understand the reasoning behind the performance actions either on a cognitive, physiological, or biomechanical level. This too searches for pattern, however, focuses on behavioural patterns. This second approach will be referred to as Behavioural-Based Modelling.

Before presenting in detail the two approaches, it is vital to clarify what we are trying to model, which is known in the literature as Expressive Music Performance (EMP).

EXPRESSIVE MUSIC PERFORMANCE (EMP)

> He put the bow to his instrument . . . and then, the first notes, bold and fiery, sang through the hall. At once the spell began to work. Was this really the music of a violin? What grandeur in these slurred notes, what absolute purity! There came roulades of double-stop harmonic notes, and a long run across four octaves, played staccato in a single stroke of the bow . . . Then came a noble, moving theme, which sounded as though a human voice was singing . . . After the seemingly endless applause had subsided, Paganini began to play the second movement. It was an adagio, and showed the virtuoso from quite a different angle. There were none of the devilish tricks that had stunned the audience during the first movement. A sublime, angelic song of great *noblesse* and simplicity touched the hearts of the listeners . . . The notes followed one another as though growing out of the instrument, and it seemed incredible ... that this wooden object was not an integral part of the man who played it, a part of his very soul . . . The audience sat as though paralysed until the rhythm of a graceful rondo changed their mood . . . an infinitely tender pizzicato accompanied the melody, and it finally soared away into a happy dance tune (Farga, 1969, pp.171-2).

Juslin (2003) used this text above, written by Nicolo Paganini in Vienna 1828, to describe many of the recurrent ideas that surround a music performance or to be more precise, an Expressive Music Performance (EMP). Observe the use of abstract terms such as: the captivating experience, the voice-like quality of certain musical instruments, the idea that music may alter a listener's mood, the close connection between music and expression of emotions, the notion that expression is embodied in the acoustic parameters of the performance, the belief that expression 'springs from the performer's very soul', the importance of the musical piece itself in shaping the

expression, and the 'devilish tricks' commonly attributed to the expressive virtuoso.

It is the expression that makes possible new and insightful interpretations of familiar works, and it is the expressive ability that makes us prefer one performer over another. Juslin (2003) continues:

> It is the expression that makes people go through all sorts of trouble to hear human performances rather than the 'dead-pan' renditions of computers (Juslin, 2003, p.274);

Note that Juslin (2003) uses reciprocally the terms 'expression' and 'human'; He also seems to be very sceptical about the ability of computer models to produce 'expressive' music performances.

In this book, we would also like to emphasize this human aspect of expressive music performances, but without prima facie denying that computers can produce 'humanised' musical performance, or at least, something that would not be distinguishable from a musical performance carried out by humans.

The term 'expression', as used in contemporary studies of music performance, refers to a well-documented systematic deviation from mechanical regularity and the nominal values notated in the score. Variations in tempo, intensity, timbre and articulation, as well as the variations in pitch known as vibrato, constitute the most important expressive characteristics of performed music (Dogantan-Dack, 2006).

We believe that other 'human' aspects should also be considered in an 'expressive' musical performance. One in particular is the focus of this book: errors. It would be incorrect to state that an EMP must contain errors to be really 'expressive'. However, it is possible to say that a 'humanised' musical performance is very likely to contain errors, given the fallibility of human nature.

Perhaps, measuring the deviations between what is written in the score and what is performed can indeed indicate what is tirelessly

referred to as 'expressivity', but it does not reveal the intention behind the actions, this is discussed in the next section.

SIMULATION-BASED MODELLING (FIRST APPROACH)

Structure-expression relationships have been formalised in computational models that apply rules to input structural descriptions of musical scores (Sundberg, 2003; Sundberg *et al.*, 1983a; Sundberg *et al.*, 1983b). In fact, measuring the deviation of the music performance from what is actually written in the score is the most common technique to quantify the 'expressiveness' of the performance.

Extensive work has been developed to identify relevant cues for musical expression in audio signals and then, with the aid of score--matching algorithms, compare these findings with the notated score. Such cues include: tempo, sound level, timing, intonation, articulation, timbre, vibrato, tone attacks, tone decays and pauses (Poepel, 2005). This approach is referred to as analysis-by-synthesis.

Another approach referred to as analysis-by-measurement takes empirical evidence directly from measurements of human expressive performances. Both approaches use the musical notation (score) as a reference to quantify deviations.

Whatever the source of the data, some computational techniques have been recurrently used in an attempt to model the expressive performance. These models serve to generalise the findings and have both a descriptive and predictive value (Widmer, 2004). Next, we present some of these techniques and models.

Rule-Based approach

Computer scientists have been using time and time again the same successful strategy: analysing the input, analysing the output, establishing the differences between them and determining various production rules that would transform the input into the desired

output. This rule-based approach has been proven to be very valuable in deterministic scenarios. For the same input, the same output is always generated.

One of the first computer software built for musical purpose, the Groove Systems (Mathews and Moore, 1970), is an example of this rule-based approach. However, in terms of modelling performance rules no other model can compete with the KTH performance rule system.

The KTH model has been in continuous expansion for over 25 years and is perhaps the most complete model for musical performance ever built. It incorporates rules for micro-level timing, metrical patterns and grooves, articulation, tonal tension, intonation, ensemble timing, and phrasing. Since 2001, the KTH model incorporates some rules to simulate inaccuracies in the motor system derived psychoacoustic experiments, involving finger tapping tasks and models of human timing proposed by Gilden and colleagues (2001; 1995).

The 'noise' rule in the KTH model consists of two distinct components (Juslin *et al.*, 2002). The first component, motor delay noise, is assumed to originate from the effectuation of each tone gesture. It is modelled using white noise added to each tone onset time and tone sound level. Thus, this component only affects the onsets individually and does not result in any long-term tempo drift. The second component, assumed to originate from an internal time-keeper clock, is modelled using 1/f (fractional Brownian Motions) noise with the overall amount dependent on the interonset intervals (IOI). The resulting deviation from the two components closely follows the just noticeable difference (JND) often referred to in perception experiments (Juslin *et al.*, 2002). Interestingly, listeners rated performances with the noise rule applied as more 'human' but not more 'musical'

Rule-based systems, although highly efficient, can be cumbersome. The complexity grows with the number of rules modelled and every new rule requires a revaluation of all others. As Friberg

(1995) and colleagues of the KTM project discovered, a musical performance is a very complex scenario and perhaps rules are not the best way to model it.

Mathematical Approach

A rather different approach is a mathematical modelling of musical performance as proposed by Mazzola (2002). The Mazzola model builds on an enormous theoretical background, namely 'mathematical music theory'. The model covers various aspects of music theory and analysis through a highly complex mathematical approach; it also involves all sorts of philosophical, semiotic, and aesthetic considerations.

The Mazzola model consists of an analysis part and a performance part. The analysis part involves computer-aided analysis tools for various aspects of the music's structure such as metre, melody, or harmony. Each of these are implemented in specific *plugins*, the so-called RUBETTEs, that assign particular weights to each note in a symbolic score.

The performance part that transforms structural features into an artificial performance is theoretically anchored in the so-called 'Stemma Theory' and 'Operator Theory' (a sort of additive rule-based structure-to-performance mapping). It iteratively modifies the 'performance vector fields', each of which controls a single expressive parameter of a synthesised performance.

Every step of the theory is explained in specific mathematical terms and with a special terminology that is greatly different from that commonly used in performance research (Beran and Mazzola, 1999; Mazzola, 2002), but unfortunately, the artificial performances produced by such models were not compared with real performance data (Widmer, 2004). Hopefully, there are more enlightening ways of modelling highly complex scenarios, such as expressive music performance.

Machine Learning Approach

In the last decade, AI techniques have seen increased use in an attempt to identify patterns and regularities in expressive music performance (Madsen and Widmer, 2006).

This method of building computational models of expressive performance uses inductive machine learning and data mining techniques to autonomously discover significant regularities in large amounts of empirical data – precisely measured performances by skilled musicians (Widmer, 2004). This is ideal when situations that are too complex to have rules 'manually' extracted from the data.

Some of these learning algorithms produce general performance rules that can be interpreted and used directly as predictive computational models. Of course, models in human or artistic domains cannot be expected to be 'correct' in the sense that their predictions will always correspond to the behaviour observed in humans (Widmer, 2004).

An example of such an approach is the PLCG system proposed by Widmer (Dixon *et al.*, 2002; 2005; Madsen and Widmer, 2006; Saunders *et al.*, 2004; Widmer, 2003; Widmer, 2004; Widmer *et al.*, 2003). In general terms the PLCG runs a series of sequential covering algorithms in parallel on the same musical data, trying to identify patterns in the note-level of parameters such as tempo, dynamics and articulation. These resultant rules are then gathered into clusters and a single rule from each cluster is used for simulating computer-generated expressive performances.

Because the extraction of the rules is automatic, it is crucial for the success of the 'learning' algorithm that the data is representative and 'clean', meaning that any error should be removed. This represents a problem when the 'error' itself is the subject of study.

Furthermore, when the rules are being generated based on deviation from the score, it is assumed that any and every deviation is intentional. There is no distinction between the cognitive and motor processes.

Despite sophisticated algorithmic learning techniques, the initial limitation of this approach persists: the reasoning behind the rule does not matter as long as the rule itself works. In the next section, we present a rather different approach, where the focus is on the understanding of the internal processes that lead to certain behaviours in musical performance.

BEHAVIOURAL-BASED MODELLING (SECOND APPROACH)

The act of interpreting, structuring, and physically realising a piece of music is a complex human activity with many facets: physical, acoustic, physiological, psychological, social, and artistic (Widmer, 2004).

According to Juslin (2003), performance expression is best conceptualised as a multi-dimensional phenomenon consisting of five primary components:

1. Generative rules that function to clarify the musical structure;
2. An emotional expression that serves to convey intended emotions to listeners;
3. Motion principles that prescribe that some aspects of the performance (e.g. timing) should be shaped in accordance with patterns of biological motion; and
4. Stylistic unintended local deviations from performance conventions.
5. Random variations that reflect human limitations concerning internal time-keeper variance and motor delays;

From these five components listed by Juslin (2003), only the first one can be investigated using the traditional approach. To investigate the other four topics, a multi-disciplinary approach is required involving areas such as psychology, musicology, and biomechanics. On this basis, computational models of music performance are often used to validate cognitive theories rather than to predict values.

In the next sections, we present some of the schools of thought behind music cognition and motor control.

Musical Structure

The notated music score is but a small part of the actual music. Not every intended nuance can be captured in the formalism of a written musical notation of Common Music Notation (CMN), and the composers are well aware of this limitation (Widmer *et al.*, 2003) consequently, interpretation of the music is left to the performer.

Performers must not only decode the symbolic information written in the music score but also interpret its 'hidden' structural content to adequately communicate the composer's ideas to the listener (Drake and Palmer, 1993).

Many findings have established a causal relationship between musical structure and patterns of performance expression (Clarke, 1988; Palmer, 1989; Sloboda, 1982). One of the most well-documented relationships is the marking of group boundaries, especially phrases, with decreases in tempo and dynamics (Henderson, 1936). Patterns of *rubato* (tempo modulations) often indicate a hierarchy of phrases, with deceleration at a boundary reflecting the depth of embedding (Shaffer and Todd, 1987; Todd, 1985; Todd, 1989a).

Naturally, performers must adopt a segmentation strategy to identify these musical structures. Perceptual studies suggest that the segmentation of a musical sequence is influenced by three accent structures: rhythmic grouping, melodic and metric accent structures.

Bean (1939) however, pointed out a human characteristic acting upon the segmentation strategy: short-term memory capacity. Good sight-readers work with effective chunking (of the score) using short-term memory (Gabrielsson, 1999).

Sight-reading is especially important in the first stage of the performance plan, that is acquiring knowledge of the music and developing preliminary ideas about how it should be performed. According to Gabrielsson (1999), it is also in this first stage that the structural analysis reveals the real meaning of the musical information. The second stage involves hard work on technical problems to establish

the spatiomotor pattern required to perform the music. The third and final stage is a fusion of the two previous stages with trial rehearsals that produce a final version of the performance.

The final version of the performance is what the musician intends to replicate in front of the live audience. Would the audience be able to perceive the expression in this performance?

Perceptual Modelling

Perceptual invariance has been studied and found in several domains of cognition, including speech (Perkell and Klatt, 1986), motor behaviour (Heuer, 1991), and object motion (Shepard, 2002). It has also been the topic of several studies in music perception honing (Honing, 2006b).

In a musical context, the perceptual model tries to predict the degree of expressive freedom a performer has in a music performance before the listener perceives a misinterpretation. The rationale behind perceptual-based models is that, in general, a performer would like the listener to recognise the original, notated music.

These models attempt to predict when, for example, the rhythm performed with some tempo and timing variations will still be recognisable as such by the listener. Pisoni (1977) found listeners to be able to distinguish temporal differences between two successive acoustic events between 500 Hz and 1.500 Hz signal at a minimum relative of 20 ms. Moore et. al. (1993) found that the ability of listeners to detect gaps in a signal was around 6 to 8 ms for signals in the range of 400 to 2.000 Hz. Other techniques have shown figures as low as 2 ms at frequencies of 8,000 Hz.

The representation and control processes that underlie people's ability to recognise, store, recall, transform and generate musical material is related to the ability to make sense of abstract structural representations from a complex multi-dimensional stimulus stream, like music or language (Sloboda, 2000).

Whilst the technical component of skilled music performance is related to the mechanisms of producing fluent outputs, the expressive component is derived from intentional variations in performance parameters chosen by the performer to influence the cognitive and aesthetic effect on the listener (Palmer, 1997).

Although most perceptual models of music performance address timing, some tackle dynamic (intensity) changes as well. A performer's intentional deviations generally correspond to change in sound level that even non-musical listeners can perceive fairly well, even when underlying acoustic changes are not identifiable (Palmer, 1997). Musical experience, nevertheless, does enhance the ability to identify interpretations and expressive aspects of performance.

Perceptual models have been the preferred approach to model expressive timing in music performance (Honing, 2006a) but this is not the only means. In addition to Perceptual Models, Kinematic Models have also been used in the domain of music cognition. The latter advocates an intimate relationship between musical motion and physical movement.

Kinematic Models

To sound natural in performance, expressive timing must conform to the principle of human movement (Honing, 2003). Todd (1992) defends the principle that performance, perception of tempo and musical dynamics are based on an internal sense of motion.

This principle reflects upon the notion that music performance and perception have their origins in the kinematic and dynamic characteristics of typical motor actions. For example, regularities observed in a sequence of foot movements during walking or running are similar to regularities observed in sequences of beats or note values when a musical performance changes tempo.

The relationship between musical motion and physical movements has been studied as a form of modelling music cognition and

expression (Todd, 1995b). It focuses on the identification of patterns that are commonly found in music performance and establishes how these patterns conform to the laws of physical motion.

A considerable amount of theoretical and empirical work attempts to illustrate apparent relations between physical motion and music (Honing, 2003) mostly by analysing the expressive timing of the last sequence of notes in a performance (final *ritard*) alluding to a runner coming to standstill (Desain and Honing, 1994; Honing, 2001; Repp, 1994; Sundberg, 1980; Todd, 1992).

A shared assumption from these works is that we experience and make sense of musical phenomena by metaphorically mapping the concepts derived from our bodily experience of the physical world into music. Accordingly, listeners hear the unfolding musical events as shaped by the action of certain musical forces that behave similarly to the forces behind our movements in the physical world such as gravity and inertia (Dogantan-Dack, 2006). Baily (1985) even argues that the performer's internal representation of music is in terms of movement, rather than sound.

Even if the 'motion' approached by these psychological studies is in the metaphorical plane, these studies borrow from mechanics and kinematics the terms that describe motion, as if it was something tangible. They talk about mass, force, and speed of an object in terms of velocity, time, and place. Some studies go even further and actually apply the laws of physics to musical events.

An example of such a literal interpretation of 'motion' is the work of Das et al. (2001). Based on the fundamental assumption, first proposed by Todd (1995b), that motion in music can be modelled using Newtonian mechanics, Das (2001) performed a statistical analysis of MIDI data and found four basic up-down motion types in music. By motion, Das (2001) meant 'a shift of tension that constrained within the dimensions of music time and space and realised through music structure'. Interestingly, it was found that tempo

variations in music performance are indeed compared with the behaviour of physical objects in the real world.

Arguments against kinematic models suggest that physical notions of energy cannot be equated with psychological concepts of musical energy (Desain and Honing, 1992). Another criticism of the kinematic models is that they are insensitive to the rhythmic structure of the musical material (Honing, 2003).

Furthermore, if performers have their own specific force and mass then it would not make sense to try to find one general curve that would apply to all performers; this would not correspond to musical reality. An overall curve shape predicted by the rules that come with human motion does not convey enough evidence to support kinematical models of expressive timing (Honing, 2003).

In summary, Honing (2003) states that the perceptual approach should be preferred over the kinematical approach in order to model to music cognition. However, he also defends an ultimate solution embracing both the cognitive and embodied aspects of music perception and performance.

Internal Time-keeping System (Motor Control)

In music performance, the motor system assumes the role of planning the upcoming movements necessary to execute the task on the basis of internal clocks. The primary role of the internal clock is to regulate and coordinate complex time series such as those produced between hands (Povel and Essens, 1985); but it also acts as timekeeper by controlling the time scale of movement trajectories (Shaffer, 1981).

Fraises (1982) suggests that our internal clock operates at a prefered rate of 600 ms at the level of *tactus*. For instance, people often generate beat patterns around 600 ms in spontaneous rhythmic tapping tasks. Periods greater than or less than this primary timing level are achieved by concatenating or dividing beat periods (Shaffer, 1981).

Naturally, most models based on internal-clocks exert their influence at the metrical level in a musical sequence (Parncutt, 1994). For instance, there is evidence that the timing of musical notes in piece changes according to different tempi in motor exercises as Gabrielsson (1999) reported:

1. Faster or slower tempi present a higher variability of inter--note intervals than intermediate tempo;
2. The velocity of the key-press (piano) increases with tempo;
3. Left and right hands present different key-press (piano) velocities, note durations, and overlap between consecutive notes.

Performance timing can also exhibit stability at more abstract hierarchical levels, such as entire musical pieces. The standard deviation of the total piece (35-40 min) duration is about 1% smaller than that of individual movements within the piece (Palmer, 1997). In simple terms, if one movement is shortened, another compensates in duration, which suggests temporal control at a level higher than the individual movements.

Motor control is responsible for planning and synchronising the movements of the musician but when it comes to physically perform the movement, biomechanical constraints take over. It is due to the muscles, joints and tendons that the performer is most exposed to failures and breakdowns either caused by internal (e.g. fatigue) or external (e.g. temperature) factors.

Biomechanical Models

Psychological studies of music performance have provided a wealth of information on musical expression and its relationship with the structure of a piece. However, these studies have largely ignored the physical manipulation of the instrument by the performer, even though the mechanics of the player's body is assumed to play a decisive role in shaping the sound (Sundberg, 2000).

Performance is traditionally the means through which works of music reach audiences, and it is the performance that makes the physicality of the body behind the music immediately evident to listeners (Dogantan-Dack, 2006). Yet, it is not common for music performance models to consider biomechanical constraints in the generation of music performances.

More often biomechanical models are used in the understanding and prevention of possible injuries resultant of the accumulation of micro traumas when the human physiological limits are exceeded, a common problem for musicians (Ericsson, 1993). Nevertheless, two studies that contemplated the properties of the human body in the modelling of musical performance have competed:

1. An ergonomic model for piano fingering (Parncutt *et al.*, 1997);
2. On the complexity of classical guitar (Heijink and Meulenbroek, 2002);

Ergonomic Model for Piano Fingering

Could the modelling of piano performance, like *timbral* synthesis, benefit from the introduction of physical models of the performers' bodies, brains, ears, lungs, lips, and fingers? This was the fundamental question Parncutt (1997) was trying to answer when he implemented a 'virtual pianist' that incorporates an ergonomic model for fingering.

As in the case of guitar performance, piano performance accommodates a number of different fingerings for a given configuration of notes. There is no such thing as a 'standard' fingering for given notes, although Parncutt and colleagues (1997) have noted that fingers 2 and 3 are more often used than fingers 1, 4 and 5. The fingerings used by keyboard players are determined by a range of ergonomic (anatomic/motor), cognitive, and music-interpretative constraints.

Based on biomechanical findings extracted from the literature (no experiments were reported), Parncutt (1997) designed a set of rules that would assign points to the most suitable fingering within

a particular musical context. The fingering gaining the most points was then chosen as the optimum pattern.

Basically, Parncutt's rules considered the stretch of the fingers, displacements, the use of weak fingers (4 and 5) and the thumb. The rules were based on Piano playing techniques and, as a consequence, not portable to any other instruments. However, some of his ideas and his approach to the problem are relevant to this work. For instance, Parncutt realised that the application of raw biomechanical models in musical performance may be possible but it is not adequate. Instead, a distinction between variables like *Maximum Possible Span* (hand's anatomy) and *Maximum Practical Span* (instrument playing technique) is necessary.

Although Parncutt's model did manage to predict some of the fingering choices and avoidances when confronted to the fingering preferences of human pianists, his results are questionable because his model ignored crucial cognitive aspects, such as the use of common fingerings for scales and arpeggio, rhythm, tempo, articulation, register and style. More recently, Jacobs (2001) identified some drawbacks in Parncutt's model and successfully proposed some refinements, most of which were related to the weak-finger rules and a new scoring system based on physical distance range. Unfortunately, no addition regard to the cognitive side was made.

On the Complexity of Classical Guitar

Heijink and Meulenbroek (2002) conducted a behavioural study to explore the biomechanical basis of the complexity of the left-hand movement in guitar playing. Three factors were analysed in relation to the notions of postural comfort when playing a sequence of single notes:

1. The position of the left hand on the guitar neck;
2. Finger span;
3. Hand repositioning;

Their study protocol resorted in a performance-related definition of travel-cost of a movement, proposed by Rosenbaum (Rosenbaum, 1995), which assumes that a guitarist is likely to choose the fingering that requires the least amount of physical effort when no other overriding cognitive or musical constraints need to be taken into account. This study has high relevance to the present work and whenever appropriate we will compare our results.

Based on the findings of Heijink and Meulenbroek (2002) work, Radicioni and Lombardo (2005), Tuohy and Potter (Tuohy and Potter, 2005a), and Radisavljevic and Driessen (2004a; 2004b) implemented guitar fingering models. Fingering is a cognitive process that maps each note on a music score to a fingered position on some instrument. It is also one of the most fertile areas of study when modelling guitar performance.

Several approaches have been adopted in the attempt to find the 'best' fingering to perform a musical piece on the guitar. All of them implement some form of a biomechanical cost function to decide between alternative fingerings, irrespective of the computational technique used (Burns and Wanderley, 2006; Naofumi Aoki, 2004; Radicioni *et al.*, 2004; Radicioni and Lombardo, 2005; Radicioni and Lombardo; Radisavljevic and Driessen, 2004a; Sayegh, 1989; Tuohy and Potter).

While Radicioni and Lombardo (2005) opted to solve the problem using a graph search algorithm, Tuohy (2005b; Tuohy, 2006) experimented with neural networks and genetic algorithms. However, both of them have used similar movement cost functions based on the vertical and horizontal repositioning of the left-hand, finger displacements, and finger span, and guitar fretboard region; all of those variables were measured by Heijink and Meulenbroek (2002).

A similar approach has been previously tested by this author in an agent-based guitar performance system (Costalonga and Viccari, 2004; Costalonga *et al.*, 2008). In this work, we had the opportunity to test the argument that we wish to put forward here that

biomechanical factors play a secondary role in the performer's choice of fingerings. Indeed, biomechanical constraints do limit the available options of possible fingerings, however musical style, personal preference, and other cognitive factors are more pertinent than biomechanical, as also observed by Heijink and Meulenbroek (2002).

The evidence that other attributes might play a more decisive role in the selection of fingering does not diminish the relevance of biomechanical modelling. On the contrary, it suggests that the biomechanical approach should be used when modelling involuntary movements, or in conjunction with other music performance modelling approach.

THE ROLE OF ERRORS IN A MUSIC PERFORMANCE

Performers do have an impressive ability to replicate the expressive profile of a piece in performance, with a degree of variability in the timing properties of a performance of one per cent or less (Clarke, 1993). However, even expert performers will eventually err for a variety of reasons (Palmer and Van de Sande, 1993).

Deviations from the musical notation are expected in Western tonal music as part of a performer's artistic license, and it is often difficult to distinguish these artistic deviations from actual errors (Palmer and Van de Sande, 1993). In fact, errors can lead to unexpected musical discoveries that ultimately improve the performer's technique and, as a result, enrich the performance; this effect is known as serendipity.

The problem of distinguishing deviation from errors was first noted by Desain et. al. (1997) while he was attempting to produce a more robust score-matching algorithm. Desain (1997) mentioned three situations that led score-matching algorithms to perform poorly:

1. There may be events in the score that are not written out completely (e.g., certain kinds of ornaments);

2. In the case of parallel voices, expressive timing may cause the order of events in the performance to be different from the order specified in the score;
3. Finally, the performer could omit, insert or change notes by mistake, often resulting in many alternative interpretations, especially in the case of repeated notes of the same pitch.

As Desain (1997) observed, performers never play equally. In all human performance tasks, errors seem to be a frequent occurrence and they come from different sources: cognitive, motor or mechanical (Palmer and Van de Sande, 1993).

Although errors are a frequent occurrence in music performance, there is little documented evidence of this (Palmer, 1992). Perhaps the most influential study of error in music performance is the work of Palmer and Van de Sande (1993); Nevertheless, it is a study of psychology that aims to investigate cognitive plans of music performance; for that reason, motor and biomechanical constraints are not contemplated.

According to Wickens and Hollands (2000, p.495), errors can be classified as: mistakes, slips and lapses. Errors of interpretation are called mistakes and originate from cognitive processes. Slips are quite different from mistakes, in a slip the understanding of the situation is correct and the correct intention is formulated, but the wrong action is accidentally triggered due to a motor or biomechanical problem. Lapses overlap these categories; they are the failure to perform an action when a procedural step is missing which could originate at the cognitive, motor or mechanical level.

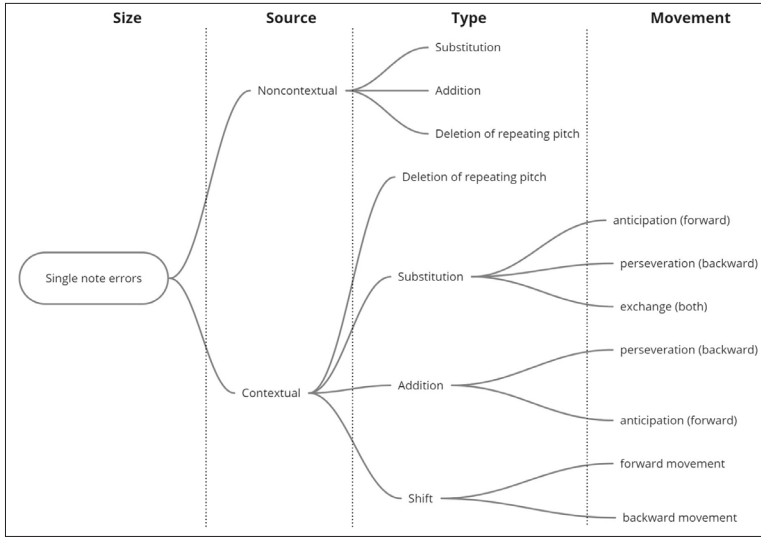
Cognitive Errors (Mistakes)

In the field of psychology, there is a belief that errors in skilled performance arise due to multiple internal representations of the desired behaviour (Garrett, 1980; Norman, 1981). Articulatory properties (motor commands produced for a specified sequence of successive

events) are believed to be a secondary cause in error production, merely influencing performance plans. Nevertheless, it is acknowledged that the mental plans underlying music performance must consider production constraints in addition to perceptual constraints. For instance, keyboard performances of musical scales suggest a greater range of articulatory control for the right hand than for the left hand (MacKenzie and Van Eerd, 1990).

In their investigation of the cognitive errors in music performance, Palmer and Van de Sande (1993) adopted a similar error coding scheme to that used in speech error research (Dell, 1986), adapted for the musical domain. A part of this coding scheme is shown in Figure 2.

Figure 2: Categories of 'production errors.



Source: Redesigned image inspired by Palmer and Van de Sande, 1993, p.459.

The classification presented in Figure 2 only considers pitch errors. The 'source' indicates whether the classification of the error

considered the surrounding musical context or not. The error types were: note addition, note deletion, note substitution, and note shift.

Substitution involves a *note event* replacing a target; an addition involves a *note event* being added (without replacing a target); a deletion involves a target being deleted, and a shift involves the movement of a target to a neighbouring location. Finally, contextual errors can reflect the range of influence of different plans in the type of movement, including forward movement (an event performed too early; anticipations), backward movement (an event performed too late; perseverations), or both (events switching neighbouring locations; exchanges).

The results reported by Palmer and Van de Sande (1993) show that most errors (98%) involved one size unit (chord or note) and most errors (91%) involved single-notes (whether from part of a chord or a solitary notated event). Contextual errors made up 57% of the total errors, the greatest percentages of which were substitutions (31%) and contextual deletions (deletion of a repeating pitch, 31%). Of the movement errors (substitutions, additions, and shifts, which comprised 69% of the contextual errors), the forward (early) movement was most frequent (52%), backward (late) movement second most frequent (37%), and bidirectional movement (exchanges) least frequent (11%).

The 'production errors', as referred by Palmer and Van de Sande (1993), indicated different influences of conceptual (melody interpretation), compositional (across- and within-voice associations), and articulatory processes (hand and finger movements) in planning music performance. In addition, the size, harmonic dimension and diatonic dimension of production errors suggest that retrieval of musical elements from memory reflects multiple structural levels and units (Palmer and Van de Sande, 1993).

Palmer and Van de Sande (1993) also reported that articulatory advantages are independent of conceptual processes of interpretation. Evidence shows a reduced likelihood of error in the highest

frequency voice, which are normally controlled by outer right-hand fingers; the authors accredited this fact to a consistent and well-learned mapping of the melody to outer right-hand finger movements in keyboard performance. Nevertheless, the authors also acknowledge to ergonomic and biomechanical implications in such behaviour.

Human Factor Engineering, Ergonomics, and Biomechanics Errors (Slips)

Before the birth of human factors or ergonomics, the emphasis was placed on 'designing the human to fit the machine' (Wickens and Hollands, 2000). Therefore, it is not unusual to find performers contorting themselves around musical instruments that were designed in the last century, when ergonomics and human factors were not formally taken into consideration when building a musical instrument.

Meister (1989) defines human factors as 'the study of how humans accomplish work-related tasks in the context of human--machine system operation, and how behavioural and non-behavioural variables can affect that accomplishment' (Meister, 1989, p.2). Ergonomics is a broader scientific discipline concerned with the interaction between humans and artefacts (Salvendy, 1987, p.3).

Both the Ergonomics and Human Factors fields have been concerned to understand the limitation of human abilities independently of its source, be that cognitive, motor, or biomechanical. The fundamental goal is to reduce error, increase productivity, and enhance safety and comfort when the human interacts with an artefact or system (Wickens and Hollands, 2000).

There have been several attempts to classify and model the types of errors that people make during a task to predict and avoid them. A well-known technique used in the human factors field to analyse error is the THERP - Technique for Human Error Rate Prediction (Swain *et al.*, 1983). THERP provides extensive guidelines for an analyst to identify errors that might occur at each point in a task

analysis and assign probabilities to each error. More on the behavioural side, SHERPA - Systematic Human Error Reduction and Prediction Approach (Embry, 1986) specifies potential psychological error mechanisms and then identifies the resultant error.

A more practical approach was proposed by Reason (1987) with GEMS - Generic Error Modelling System. GEMS focus in on the rule-based and knowledge-based behaviours and it has been used to analyse the errors in a variety of industrial situations. Both SHERPA and GEMS are based on Rasmussen's SRK model (Rasmussen, 1986).

Extensive research has been done to understand the causes of errors at the cognitive, motor and mechanical level, but unfortunately few studies have targeted music performance. Conversely, most of the motor control studies in music performance do recognise the relevance of the error (Haslinger *et al.*, 2004; Juslin, 2003; Repp, 2006; Sloboda, 2000).

In order to exemplify the relevance that errors might have in a music performance context, we can compare it with the findings of a similar motor task: typing in a word processor. Card and colleagues (1983) estimated that the typists make mistakes or choose inefficient commands on 30% HEP. The human error probability (HEP) is the basic unit of human reliability in discrete tasks and it is estimated from the ratio of errors committed to the total number of opportunities for that error (Freivalds, 2004).

The challenge is to establish when errors are caused by cognitive processes and when they are caused by mechanical and motor limitations of the body. Is the music piece demanding more than is humanely possible? If so, what are the consequences? In this book, we try to address some of these questions in the context of guitar performance.

SOUNDING REALISTIC

So far, we have discussed the implications of the internal and motor processes of the performer in the music performance. However, even

if all of these processes were already fully understood and could be formalised in computer models, we would still need to consider the characteristics of the musical instrument.

There are two main approaches to producing the sounds of a musical instrument played by an 'artificial' performer: a) build a robot that is able to play the real acoustic instrument; or b) model the real acoustic instrument on a computer.

Bilitski (2005) opted for the first approach and built an artificial mouth that is used to 'play' (produce notes) the trumpet. The primary use of an artificial mouth is to develop physical models of the human mouth and to enable automatic music performance of wind instruments. Similar strategies have been used with bamboo flute (Mizuno and Takashima, 2001) and brass (Gilbert *et al.*, 1998).

In essence, an artificial mouth consists of a flux of air passing through an elastic substance that vibrates resembling human lips. The challenge of such an approach is to make the machine behave in the same way as a human body and, in this case, having to determine how much lip pressure and air pressure to use to play each note. To do so, Bilitski successfully used a Genetic Algorithms (GA) approach. Interestingly, the termination condition of the adopted fitness function of the GA was a variation in pitch of +/- 0.4%. This is because the human ear cannot detect this pitch difference. Once again, a human limitation contributes to the modelling of music performance

Although interesting, this first approach is highly complex. Fortunately, it is not necessary to wait for the advances in robotic hands in order to artificially produce guitar performances. More commonly, guitar sounds are generated by Sound Generation Units, such as synthesisers and samplers. The advantage of this second approach is that these Sound Generation Units 'speak' directly with the computer models of music performance through well-established communication protocol for digital instruments.

Unfortunately, the standard technology for interfacing digital musical instruments (MIDI) was designed for keyboard instruments

(Poepel, 2004a). Consequently, any performance idiosyncrasies of other types of musical instruments may not be supported. If the 'bridge' between the Performance Models and the Sound Generator Units does not allow the traffic of specialised performance messages, then the acoustical realism of the synthesizer may have little importance because it cannot be controlled expressively.

The technical limitations of modern synthesisers can be explained on the basis that not enough investigation which aims to understand the biomechanics involved in a guitar performance has been done. Commercially, the development of such synthesising technology is not viable because no one would know how to use it. As a result, Sound Generator Units (synthesizers and/or samplers) tend to produce a deterministic and inexpressive simulation of music performance that does not consider the human aspects involved in the task of performing a musical instrument.

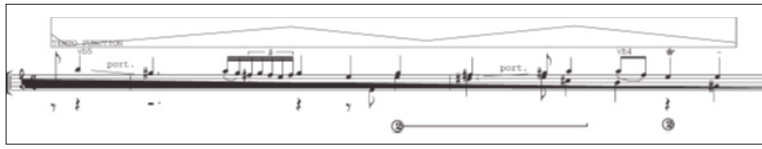
In a guitar performance, for instance, information such as different pluck styles (shape, position, angle etc.), vibrato, and dynamic variations need to be embedded in the model (Karjalainen *et al.*, 1993). Also, special effects such as rubbing and scraping of the string and various knockings on the guitar body are essential in synthesizing modern guitar repertoire (Erkut *et al.*, 2000; Tolonen, 1998; Valimaki *et al.*, 1996).

The real-time software synthesizer called PWSynth is an attempt to effectively integrate guitar performance techniques and sound synthesis proposed by Laurson *et al.* (2001). PwSynth is a library for PacthWork (Laurson 1996) that implemented a physical model synthesis of an acoustic guitar. To control the synthesizer a music notation software package called ENP – Expressive Notation Package is used.

The use of an 'expressive' notation is necessary because, as seen in section 2.3.1 (p.31), the Common Music Notation (CMN) does not specify all the features of music performance. The ENP extends the CMN allowing the user to enter both standard and non-standard expressions specific to an instrument and playing style.

The expressions in ENP can be applied to a single note (such as string number, pluck position, vibrato, or dynamics) or a group of notes (e.g. left-hand slurs or finger-pedals). Macro expressions generate additional note events, such as tremolo, trills, *portamento*, and *rasgueado* (a strumming technique using the right-hand fingernails that is common in the flamenco playing style). ENP also allows fine-tuning of timing with the help of graphical tempo functions.

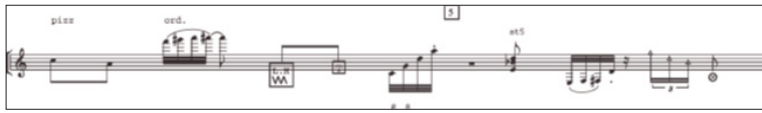
Figure 3: ENP Example 1.
An expressive notation (ENP) example showing a transcription of 'Loure from the E major Partita for lute' by J. S. Bach.



Source: Redesigned image inspired by Laurson et al., 2001, p.44.

Figure 3 shows an ENP example from the classical guitar repertoire Note that in addition to conventional pitch and rhythm information, the score contains several standard expressions, such as left-hand slurs and the encircled '2', which indicates that the corresponding notes should be played on the second string. Furthermore, there are several *portamento* expressions (lines marked with 'port') that indicate a rapid glide of the left-hand finger. The non-standard expressions '*vb5*' and '*vb4*' denote that the notes in question should be played with a moderate vibrato (Laurson et al., 2001, p.44)..

Figure 4: ENP Example 2.
Continuation of the previous ENP example shown in Figure 3.



Source: Redesigned image inspired by Laurson et al., 2001, p.45.

The ENP also allows the use of non-standard *note heads*, which in turn permits the user to express novel instrumental playing techniques. For instance, the first non-standard *note head* (the box with a triangularly shaped waveform right after the first run is shown in Figure 4) indicates that the performer should rub the strings with the left hand. The second one (the small box containing the letter 'T') stands for a *tambura* effect whereby the player hits the bridge of the instrument with the right-hand thumb. The last one (a note-head with an encircled 'x') indicates a hit with the right-hand nail on the body (*golpe* in the Spanish terminology).

The combination of PWSythn and ENP is a rather good attempt to integrate expressive control and synthesis but it does have a drawback. All the performance actions (controlling messages) must be specified by the user. Even with all the flexibility that the ENP supports, a specification of a truly realist performance would be extremely time-consuming. Furthermore, the user would have to be an expert guitarist with a high sense of self-awareness regarding all the actions he produces during the performance. Ideally, the ENP score should be derived from a guitar performance model and this is another issue that is addressed in this book.

SUMMARY

Music performance provides a rich domain for the study of both cognitive and motor skills. It is a non-verbal communication between the composer, the performer, and the listener. The composer and the listener have been psychologically studied in order to understand the impact of the music on the listener. The performer is the link between the composer and the listener and the main focus in the expressive music performance modelling.

The performer's interpretation of the music is transmitted to the audience through special techniques that aim to highlight the composer's ideas and to convey emotion. The performer is mainly

studied from the cognitive side, but he can also be studied from the motor and mechanical perspectives.

As in any other human activity, music performance is subject to errors. These errors can be cognitive, motor or mechanical. Normally, errors are not included in music performance models despite being a common occurrence. This is partly because machine learning algorithms do not handle errors very well when searching for patterns in performance data.

Several computational approaches have been used to model expressive musical performance: rule-based systems, mathematical and statistical models, case-based reasoning and, more recently, machine learning. The only work that actually implemented 'noises' from the motor control as part of music performance is KTH. In a perceptual study of the KTH model, listeners reported having found computer performances with errors more 'human' but not more musical.

Another important aspect of a realist computer-generated music performance is the sonority produced. To be able to recreate the sounds of performance actions, a synthesiser must allow expressive control of less non-standard musical techniques (e.g. finger rubbing on the guitar's string) and even unwanted noises (e.g. muffled notes or buzzed-notes).

Chapter 3

The Multiple Aspects of Guitar Performance

> The guitar must accommodate itself to the body, not the body to the guitar. (Carlevaro, 1984, p.2)

In this chapter, we analyse the different elements involved in a guitar performance including the playability of the guitar and biomechanical constraints of the guitarist.

The chapter is divided into three main parts. In the first part, we look into the mechanics of guitars. In the second part, we present an overview of guitar playing techniques and how the human body conforms to this task. The third and last part discusses some common performance errors and their causes. The chapter concludes with an application of these aspects within the context of this work.

THE MECHANICS OF THE GUITAR

The guitar has established itself during the 20th century as the world's most popular musical instrument (Bennett and Dawe, 2001). Over

the last eighty years, it has become the favourite type of instrumental accompaniment for both artists and groups. It is estimated that today there are over fifty million guitarists around the world (Chapman, 1994).

This popularity and the stylistic development that the guitar brought to some musical genres (e.g. flamenco, country, blues, etc.) led to the emergence of many different playing techniques. This has made the guitar a highly expressive instrument (Poepel, 2004b) and consequently made computer modelling of the guitar performance a complex task.

The difficulty in simulating a realistic guitar performance by computer starts with the limited guitar synthesis techniques available. Synthesis techniques based on physical modelling consider just one of many possible configurations of the acoustic instrument (i.e. jumbo-guitar with nylon-string, spruce-top, finger-picking etc.). Any variation in the guitar setup or playing technique would require a whole new model.

Even if the guitar synthesiser adopted a more flexible approach towards rendering the performance, the problem of controlling the performance parameters would remain. In this section, we will focus on the attributes of the guitar that we believe should be modelled to produce a truly realistic computer-generated guitar performance.

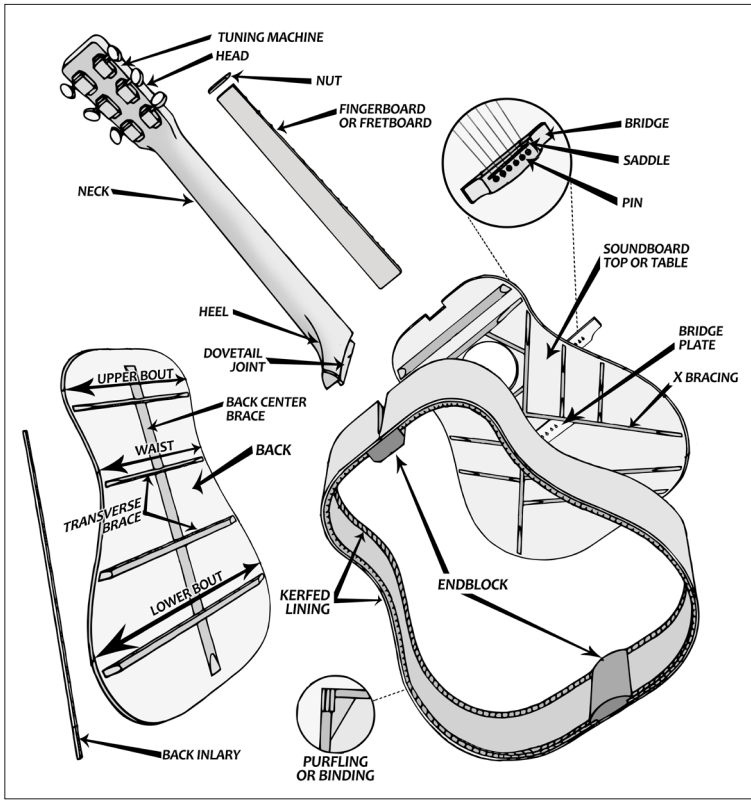
Guitar Characteristics

A guitar can be classified by its acoustics, playability, fitness and aesthetics. Playability and fitness are interlinked parameters, and they are the focus of our investigation.

Playability (responsiveness) determines how much effort is required to achieve clear, well-formed individual notes, particularly during rapid and difficult passages. Fitness focuses on how well the instrument suits the performer's characteristics to improve the instrument playability.

The guitar design and construction are surrounded by an almost mystical aura that fuses carpentry skills and art. The *luthier* can build the same type of reputation and fame as the guitarist itself. The combination of the woods, cuts, proportionality, body dimensions and many other factors determine the quality of the instrument and its suitability for the performer and musical style. It is outside the scope of this research to digress into the nuances of the *lutherie.* Rather, we would like simply to highlight the mechanical attributes of the acoustic guitar. Figure 5 shows the guitar parts that we will be referring to across this section.

Figure 5: Guitar parts.



Source: Redesigned image inspired by Evans and Evans, 1977, p.261.

The Fretboard

The fretboard is a thin long strip of wood, usually ebony or rosewood, which is laminated on the front of the neck of the guitar and above where the strings run. It has raised strips of hard material (usually metal) perpendicular to the strings against which the strings are stopped, named frets. The number of clear frets usually stays between twelve (classical acoustic guitar) and twenty-three (electric guitar). Note that the number of clear frets may differ from the total number of frets in the fretboard.

The space between the frets, known as inter-fret space, is an attribute that greatly interferes in the guitar's playability. The classical guitar technique demands a finger span of four frets. If the inter-fret is too wide then overstretching issues (injuries, delays, discomfort) are likely to occur; if too narrow and fingers may rub against neighbouring strings or frets. A general formula to calculate the inter-fret spaces is given by Equation 1 where d = distance from nut; s = scale length; n = fret number (Mottola, 2006);

Equation 1: Fret Distance

$$d = s - \left(\frac{s}{2^{\frac{n}{12}}}\right)$$

Typically, the fingerboard is a long plank with a rectangular profile. On a guitar, the fingerboard appears flat and wide, but it may be slightly curved to form a cylindrical or conical surface. Normally only classical and rhythmic (chord playing) guitars have flat fingerboards. Almost all other guitars have at least some curvature.

The flatter the fretboard, the more comfortable the guitar is for chord and rhythm playing. A curvier fretboard is more appealing to fast solo players since the curvature prevents *fretting out* (having the string rubbing against a higher fret during a bend).

Figure 6: Scalloped fingerboard.

Source: "Scalloped fingerboard (Escala escalopada)" by Celso Freire, Luthier is licensed with CC BY-NC-ND 2.0. To view a copy of this license, visit https://creativecommons.org/licenses/by-nc-nd/2.0/.

The fingerboard can also be scalloped by scooping out the wood between each of the frets to create a shallow 'U' shape as Figure 6: shows. The result is a playing surface where the players' fingers come into contact only with the strings (not touching the fingerboard) therefore creating less friction for bends and vibratos, which result in better control while playing; it also allows guitarists to play faster because they do not have to invest as much effort into fretting each note.

Playing a scalloped fingerboard requires a careful balance of pressure because too much pressure can change the pitch of the fretted note (i.e. during a bend), and too little pressure can cause fret buzz. Consequently, the majority of players choose to use a traditional fingerboard on their instruments.

Body Style and Size

Ideally, each guitar should be designed according to a guitarist's anthropometry, however, this is not economically viable, so major manufactures have established some middle-range values based on the average proportions of potential users.

Acoustic guitars come in a variety of shapes and sizes, from small travel size to jumbo and dreadnought. The body style of an acoustic guitar determines sound projection and tonal emphasis. Things to consider are tonal quality vs. playing comfort. For example, some acoustic guitar bodies come in a single-cutaway design like the shape of the Gibson Les Paul guitar, facilitating the access to the higher frets.

Although the criteria to choose the guitar's size and style are usually based on acoustical preference, it is also important to consider the fitness of the instrument to the guitarist. Furthermore, the style of the guitar body is also linked to the musical style, its role in a musical performance (e.g. lead, rhythm, solo), and the arpeggio technique (e.g. *flatpick*, *fingerstyle*). Selecting the wrong guitar for a musical style may increase the probability of errors in the performance.

The guitar's body style and dimensions vary according to the manufacturer; the dimensions used for the Martin guitars are presented in Table 1. Figure 7 illustrates the attributes presented in the data.

Figure 7: Guitar's body measurements

Source: Own image.

Table 1: Martin guitars dimensions (inches).

Type	Clear Frets	Scale Length	Length	Upper Width	Lower Width	Depth (max.)
Concert	12	24.9	19 – 1/8	9 – 1/2	13 – 1/2	4 – 3/16
Grand Concert	12	24.9	19 – 5/8	9 – 3/4	14 – 1/8	4 – 1/16
Auditorium	12	24.9	20 – 7/16	10 – 3/4	15	4 – 1/16
Grand Auditorium	14	25.4	20 – 1/8	11 – 11/16	16	4 – 1/8
Orchestra	14	25.4	19 – 3/8	11 – 1/4	15	4 – 1/8
Dreadnought	12	25.4	20 – 15/16	11 – 1/2	15 – 5/8	4 – 3/4

From the playability perspective, one of the first attributes to be observed regarded to the guitar body dimension is scale length; as previously seen Equation 1, the scale length has a direct influence on

the inter-fret spacing and string tension. For the acoustical perspective, the scale length is usually related to the loudness of the guitar.

String Action

String action refers to the distance between the strings and the frets. Even a small variation on the string's height affects the playability and tone loudness. The action is determined by the neck angle and nut/ bridge heights.

The neck angle (also referred to as the neck's pitch, tilt, or inclination) sets not only the string action but also the string intonation. This is because it affects the amount of stress that is being put on the guitar through the strings, contributing to the loudness of the tone and the longevity of the structure.

The action (along with tension) determines the amount of pressure that is necessary to depress the strings against the fretboard (stop the string). High action is not necessarily a negative characteristic of the guitar as observed in Table 2. A suggested reference value for string action at the 12th fret is 4 mm (1/8-inch) (LeVan, 2005).

Table 2: The characteristics of high and low string actions.

	High Action	**Low Action**
Pros	Loud volume; Sharp tones;	Lighter to press; Easy to play a fast run;
Cons	Requires more force; Hard to play a fast run;	Softer volume; More likely to happen to 'buzz';

String Gauge and Tension

The string gauge represents the ability of a string to retain its 'memory' over time. Heavy strings can hold their tuning for longer. Also, they can produce a better tone and higher volumes than lighter strings; however, they are also harder to press down and less comfortable to play.

Every guitar is designed to operate within a range of string gauges. Increasing the string gauge also increases the tension and consequently the string action, eventually increasing the stress in the structure as a whole. Decreasing the string gauge has the opposite effect but the reduction in tension may leave the neck too straight to play cleanly.

The scale length and the string gauge are key when determining the tension under which the guitar will operate. If identical string gauges are used in a short-scale guitar (22-22.5 inches from the bridge to the nut) and a long scale guitar (24-24.5 inches), the string tension on the longer scale instrument will be noticeably greater.

Typical string tensions are about 12-16 kg for an acoustic guitar and about 7-8 kg for electric guitar. Tensions above 17 Kg may either break the string or damage the guitar, if not both. Equation 2 is used to calculate the string tension, where f is the frequency of the note in Hertz, L is the scale length in metres, T is the tension in Newton, and μ is the mass per unit length of the string (Mottola, 2006).

Equation 2: String tension.

$$T = 4L^2 \mu f^2$$

As an example, suppose an arch-top acoustic guitar with a scale length of 650 mm (25.5 in) and equipped with a set extra-light bronze strings tuned in the standard 'E (1^{st} string) – B – G – D – A - E (6^{th} string)'. Using Equation 2 we calculated the tension of the strings as seen in Table 3.

Table 3: Calculated string tensions for the standard tuning.
where, Hz = Hertz, mm = millimetres, kg/m = kilogram per meter, m = meter and kgf = kilogram force.

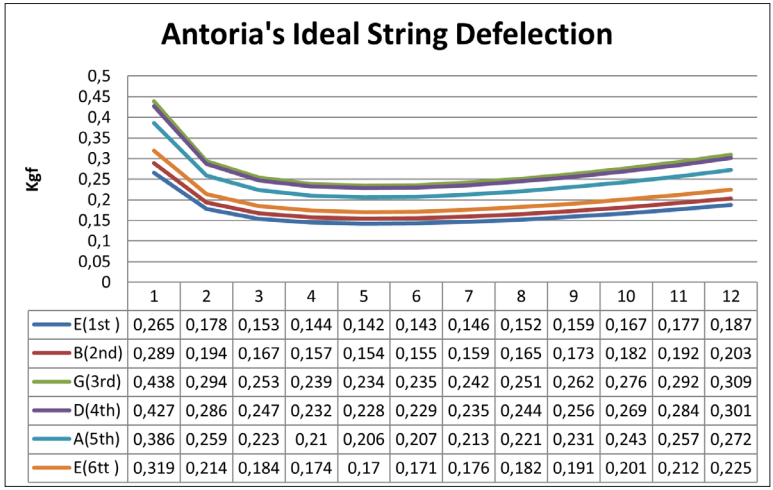
Note	Freq. (Hz)	Gauge (mm)	Mass (kg/m)	Length (m)	Tension (Kgf)
E	329.63	0.25	0.00040501	0.65	7.60
B	246.94	0.36	0.000784753	0.65	8.26
G	196	0.58	0.001892026	0.65	12.55
D	146.82	0.76	0.003285326	0.65	12.23
A	110	0.99	0.005289196	0.65	11.05
E	82.407	.19	0.007784877	0.65	9.13
Total Tension					**60.80**

The guitar in question is an Antoria Archtop Jazz Guitar equipped with a set of strings D'Addario extra-light tuned (standard) with the aid of an Intelli Chromatic Turner IMT-500. Based on the Antoria's scale length, string tensions (Table 3), and proposed string action values (Table 4) we calculated the necessary force to displace the strings for the exact length of the string action (Figure 8).

Table 4: Reference string action per fret.

Fret	Action (mm)
1	1.25
2	1.50
3	1.75
4	2.00
5	2.25
6	2.50
7	2.75
8	3.00
9	3.25
10	3.50
11	3.75
12	4.00

Figure 8: Force to displace the string towards the fretboard.
The x-axis corresponds to the fret region and y-axis to the force (kilogram-force) to deflect the string considering an ideal String Action as shown in Table 4.



	1	2	3	4	5	6	7	8	9	10	11	12
E(1st)	0,265	0,178	0,153	0,144	0,142	0,143	0,146	0,152	0,159	0,167	0,177	0,187
B(2nd)	0,289	0,194	0,167	0,157	0,154	0,155	0,159	0,165	0,173	0,182	0,192	0,203
G(3rd)	0,438	0,294	0,253	0,239	0,234	0,235	0,242	0,251	0,262	0,276	0,292	0,309
D(4th)	0,427	0,286	0,247	0,232	0,228	0,229	0,235	0,244	0,256	0,269	0,284	0,301
A(5th)	0,386	0,259	0,223	0,21	0,206	0,207	0,213	0,221	0,231	0,243	0,257	0,272
E(6tt)	0,319	0,214	0,184	0,174	0,17	0,171	0,176	0,182	0,191	0,201	0,212	0,225

Source: Own image.

The formula used in the calculation is shown at Equation 3, where *f = force, d = displacement, T = tension, and L = scale length (Hago, 2009).* It applies only for string excursions that are modest in amplitude. Also, it only considers that the plucking point will happen in half of the string length. However, a string gets progressively harder to deform nearer its extremities because the restoring force grows when the two portions of the string are unequal in length, so the Equation 3 had to be adjusted to work with the different plucking points. For example, plucking 1/4 of the way along would be calculated as 4dT/L + 4dT/3L = 5.33dT/L;

Equation 3: String deflection formula.

$$f = \frac{4dT}{L}$$

The calculations show that the average force required to deflecting the string is around 0.223 kgf. The position that required the least force (0.141 kgf) was the 5th fret, 1st string; and the position that required the greatest force (0.438 kgf) was the 1st fret, 3rd string.

Guitar Noises

The distinct tone of a guitar is the resultant sound produced by its different parts. While the top plate contributes with a very high frequency and harmonically 'simple' sound, the neck works in the middle frequency. The backplate and the ribs have the tone regarding as having the 'colour' - middle frequencies added after a time delay (Meyer, 1983).

Without a doubt, the acoustic properties of guitars play a very significant role, if not the most significant, in the simulation by computers of realistic music performance. However, the guitar's acoustic properties are not the scope of this book, except for a particular type of sound: noises. The word 'noise' usually refers to an unwanted sound, which could arguably be considered unfair. Indeed, there is an obvious relationship between noise and error; and naturally, performers do try to avoid errors. Performance errors (slips) most certainly terminate in one or another form of noise. However, we believe that is through the perception of the noise that the audience identifies the imperfect human nature behind music performance; thus noises should also be part of a computer-generated performance, but not all noise is useful. It is important to establish the right balance between noises and pure sounds. It is not any 'random' noise that will produce the desired 'human-feel' in a computer-generated performance. To do so, the correlation between the specific performance errors and the noises they produce must be found.

Noise is also an important part of the sound signature of an instrument. For instance, the sound from the finger sliding along the guitar before it is plucked is very characteristic of the guitar. It does

not happen on a harpsichord or piano performance. In addition, it contains not only noise, but also part of the later overtone spectrum of the tone, or the *eigenvalues*[1] of the guitar body. If the finger noise is left out an important part of the tone is missing (Cuzzucoli and Lombardo, 1999).

In guitar performances, there is yet another characteristic noise caused by the fingers rubbing along the string, known as pre-scratch. Pre-scratch is a term used to refer to the sound component that precedes the actual tone. It is caused by the fingers of the right-hand rubbing along the string before it is released. In the *apoyando* and *tirandu* techniques, the finger is normally placed on the strings in such a way that both the fingernail and the fingertip touch the string, producing noise of very short duration, lasting somewhere between 1 ms and 5 ms, just long enough to be audibly detectable (Valimaki *et al.*, 1996).

The 'finger slide' and the 'pre-stretches' are the type of noises that can be found in the modern physical modelling synthesis techniques (Cuzzucoli and Lombardo, 1999; Valimaki *et al.*, 1996). However, in order to make these noises sound realistic in a musical context, the moment they are used needs to be carefully selected. Finding the moment when noises (or errors) are likely to happen still demands more investigation.

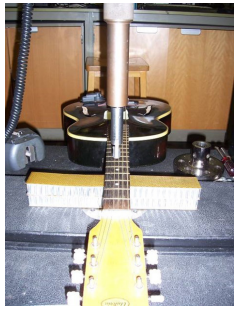
There are other noises, however, that has been largely ignored by Sound Generation Units, even though they occur consistently in guitar performances. In fact, several different noises can be produced if not enough pressure is applied to stop the string properly.

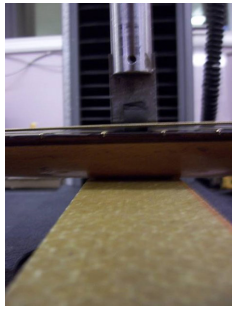
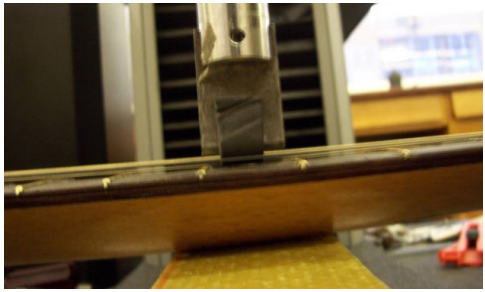
With the aid of an INSTRON 5582 Universal Test Machine (Instron, 2009) we analysed two categories of errors for which lack of force is believed to be a cause: a) muffled/damped notes; and b) buzzed notes; Figure 9 shows some pictures of the measurements being carried out.

---

1 Any of the possible values of a quantity derived from a differential or integral equation having solutions that satisfy certain special conditions

Figure 9: Pictures of the Antoria's string tension measurement.
These pictures were taken during the experiment to measure the real force that is required to produce a clean note on the Antoria guitar.

Source: Own image.

The experiment was intended to find the force boundaries needed to produce clean, buzzed, and muffled notes. All the six strings of the guitar were measured in three regions of the fretboard: 1st - 3rd frets, the 5th fret; and 12th fret. The values for the other frets were interpolated. The analysis of the quality of the note generated was subjective to the personal evaluation of the experimenter.

A muffled note can be caused by the fingertip pulp (skin *viscoelasticity*) absorbing and damping the string's vibration. To simulate this effect, the mechanical part that touches the string was equipped with a neoprene strip 5 mm thick. (Note that we are not stating that neoprene is a good material to simulate the human skin, it is purely an inexpensive and highly available material which have absorbing properties suitable to this experiment).

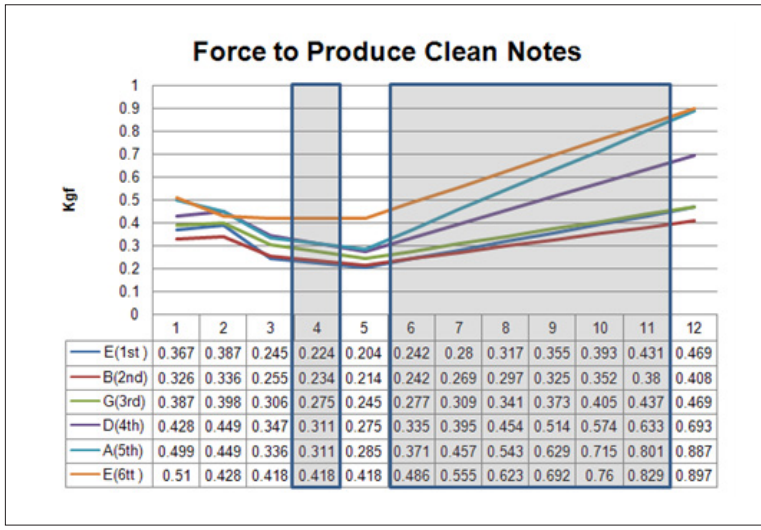
The harder the string is plucked, the greater the intensity of the vibration the material must absorb. Hence, we kept fixed the plucking force at approximately 3 Newton (0.3 kgf) and the plucking point at ¼ of the scale length (16.625 cm).

Buzzed-notes are usually generated by positioning the finger too far left on the fret but it could also be generated by the lack of sufficient pressure to fully stop the string. To eliminate the possibility of the former situation, we kept the contact point with the force cell arm within 4 mm from the fret.

Figure 10 shows the recorded values. Note that the values for fret 4 and frets 6-11 were interpolated, not recorded.

Figure 10: Forces to produce a clean note.

The x-axis corresponds to the fret region and y-axis the real force (kilogram-force).



	1	2	3	4	5	6	7	8	9	10	11	12
E(1st)	0.367	0.387	0.245	0.224	0.204	0.242	0.28	0.317	0.355	0.393	0.431	0.469
B(2nd)	0.326	0.336	0.255	0.234	0.214	0.242	0.269	0.297	0.325	0.352	0.38	0.408
G(3rd)	0.387	0.398	0.306	0.275	0.245	0.277	0.309	0.341	0.373	0.405	0.437	0.469
D(4th)	0.428	0.449	0.347	0.311	0.275	0.335	0.395	0.454	0.514	0.574	0.633	0.693
A(5th)	0.499	0.449	0.336	0.311	0.285	0.371	0.457	0.543	0.629	0.715	0.801	0.887
E(6tt)	0.51	0.428	0.418	0.418	0.418	0.486	0.555	0.623	0.692	0.76	0.829	0.897

Source: Own image.

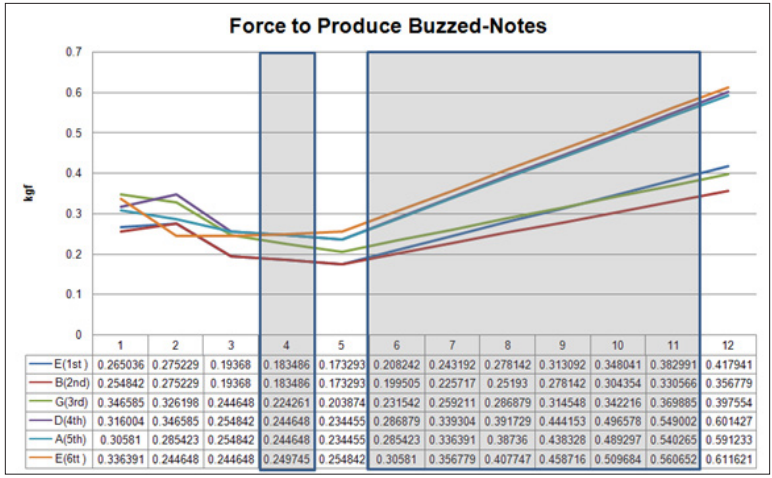
The force range required to produce clean notes stayed between 0.204 and 0.897, with an average of 0.423 kgf; this was higher than initial calculations shown in Figure 8. Surprisingly, the values recorded at the 12th fret (higher action) were even greater than the values o the 1st fret where the string is harder to deform.

The correlation of the higher recorded forces with higher string action might suggest a problem with the guitar string height adjustment. Another point to be considered is that the fingertip viscoelasticity (friction) could have a greater influence in stopping the string than anticipated. Either way, one point is clear: just the force to displace the string is not enough to stop them completely.

The measurement of the required force necessary to produce buzzed-notes adopted the same procedures used to measure the force for the production of clean notes. The results can be seen in Figure 11.

Figure 11: Forces to produce a buzzed-note.

The x-axis corresponds to the fret region and y-axis the real force (kilogram-force).



	1	2	3	4	5	6	7	8	9	10	11	12
E(1st)	0.265036	0.275229	0.19368	0.183486	0.173293	0.208242	0.243192	0.278142	0.313092	0.348041	0.382991	0.417941
B(2nd)	0.254842	0.275229	0.19368	0.183486	0.173293	0.199505	0.225717	0.25193	0.278142	0.304354	0.330566	0.356779
G(3rd)	0.346585	0.326198	0.244648	0.224261	0.203874	0.231542	0.259211	0.286879	0.314548	0.342216	0.369885	0.397554
D(4th)	0.316004	0.346585	0.254842	0.244648	0.234455	0.286879	0.339304	0.391729	0.444153	0.496578	0.549002	0.601427
A(5th)	0.30581	0.285423	0.254842	0.244648	0.234455	0.285423	0.336391	0.38736	0.438328	0.489297	0.540265	0.591233
E(6tt)	0.336391	0.244648	0.244648	0.249745	0.254842	0.30581	0.356779	0.407747	0.458716	0.509684	0.560652	0.611621

Source: Own image.

As expected, the force required to produce 'buzzed-notes' is lower than to produce clean notes, ranging from 0.173 to 0.611, on average 0.353 kgf or 0.100 kgf less. For this work, any force value below the recorded for a buzzed-note is treated as muffled-note, even though this might not always be the case in reality.

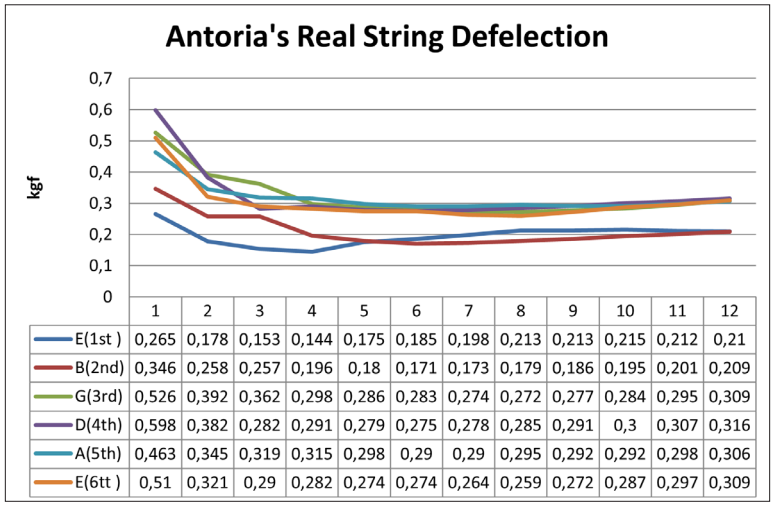
Normally, buzzed and muffled notes will often originate from a poor performance technique but it could also be due to a low-quality instrument. In order to investigate the discrepancy between the recorded and the calculated force values, we have decided to re-do the calculation using the real string action of the Antoria guitar, shown in Table 5.

Table 5: Antoria's string actions (millimetres).

Fret/String	E(1st)	B(2nd)	G(3rd)	D(4th)	A(5th)	E(6th)
1	0.175	0.15	0.15	0.175	0.15	0.2
2	0.25	0.2	0.2	0.2	0.2	0.225
3	0.275	0.27	0.25	0.2	0.25	0.275
4	0.3	0.25	0.25	0.25	0.3	0.325
5	0.30	0.26	0.28	0.28	0.33	0.36
6	0.3	0.275	0.3	0.3	0.35	0.4
7	0.36	0.30	0.31	0.33	0.38	0.41
8	0.42	0.325	0.325	0.35	0.4	0.425
9	0.44	0.35	0.34	0.37	0.41	0.46
10	0.45	0.375	0.36	0.39	0.42	0.5
11	0.45	0.39	0.38	0.41	0.44	0.53
12	0.45	0.41	0.4	0.42	0.45	0.55

As can be seen in Table 5, the string actions are slightly higher than the reference value but not enough to justify the level of difference found. Figure 12 shows the calculated force required to displace the string for the full length of the measured string actions.

Figure 12: Calculated string deflection based on real string action data.
The x-axis corresponds to the fret region and y-axis the calculated force (kilogram-force).

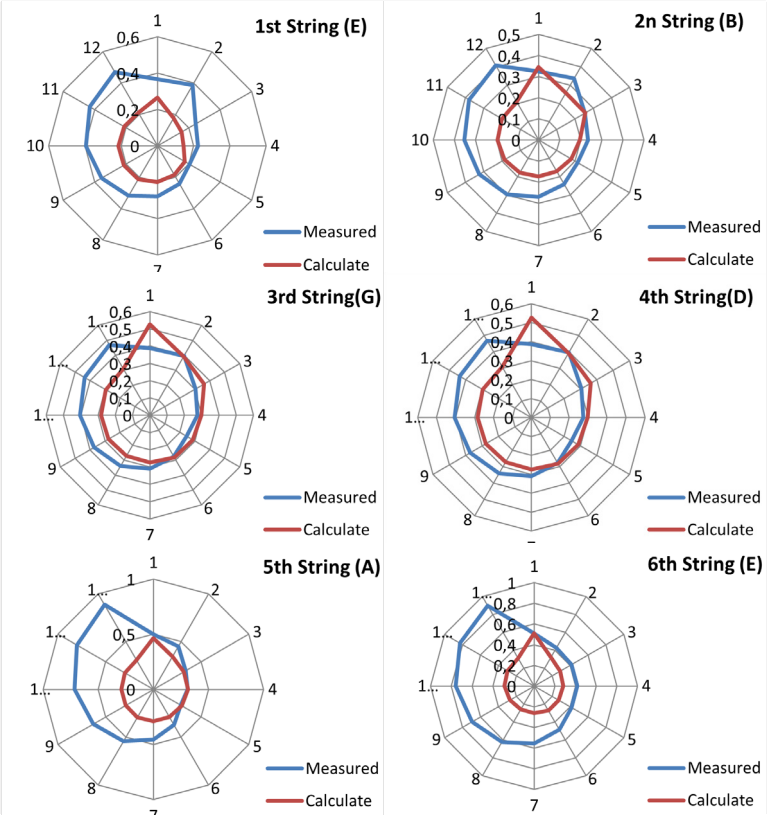


	1	2	3	4	5	6	7	8	9	10	11	12
E(1st)	0,265	0,178	0,153	0,144	0,175	0,185	0,198	0,213	0,213	0,215	0,212	0,21
B(2nd)	0,346	0,258	0,257	0,196	0,18	0,171	0,173	0,179	0,186	0,195	0,201	0,209
G(3rd)	0,526	0,392	0,362	0,298	0,286	0,283	0,274	0,272	0,277	0,284	0,295	0,309
D(4th)	0,598	0,382	0,282	0,291	0,279	0,275	0,278	0,285	0,291	0,3	0,307	0,316
A(5th)	0,463	0,345	0,319	0,315	0,298	0,29	0,29	0,295	0,292	0,292	0,298	0,306
E(6tt)	0,51	0,321	0,29	0,282	0,274	0,274	0,264	0,259	0,272	0,287	0,297	0,309

Source: Own image.

If we compare the graphs presented in Figure 8 *and* Figure 12, it is possible to observe that the string action and the instrument built quality indeed play a very significant role in the force required to displace the string. However, it did not explain the difference found in the calculated and measured force values. These differences can be visualised on the graphs presented in Figure 13.

Figure 13: Calculated vs. Measured forces for string displacement.
The outer circle (in blue) shows the measured force and the inner circle (in red) shows the calculated force. The fret regions are shown in the peripheral area of the chart.



Source: Own image.

The graphs shown in Figure 13 show the fret region values in the outer circle and the force values are represented by the inner circles. The blue line is the measured value and the red line is the theoretically calculated values. Note that for the sake of clarity the only representative frets for this comparison are the 1st, 2nd, 3rd, 5th and 12th. The others are interpolations.

As seen in Figure 13 the difference is greater in the strings with a higher gauge. The measured values are on average 0.144 kgf higher than those calculated but most of this difference comes from the 12th fret, especially on the 4th, 5th and 6th strings.

In summary, it is possible to conclude that the guitar setup and quality do play a role in the generation of buzzed and muffled notes in addition to the force. Skin viscoelasticity might also be a contributing factor deserving further investigation. The human physiological and biomechanical properties are the topics of the next section.

THE BODY BEHIND A (SKILLED) MUSIC PERFORMANCE

Skilled musicianship requires decades of regular practice, estimated at 10,000 hours (Ericsson, 1993). It is known that long-term training leads to highly stable control patterns in individuals (athletes or music performers) of professional calibre (Shan and Visentin, 2003).

Becoming skilled is largely a matter of developing new reflexes which occur without conscious control, called conditioned reflexes. Whenever a sequence of movements is practised for a long time, the complete movement pattern becomes 'engraved' in the brain, allowing the individual to shift the focus to higher-level cognitive activities, such as the interpretation of a musical piece. Skill reaches a maximum when learning has eliminated conscious control and movements have become automatic (Grandjean, 1988).

Skilled jobs call for a high degree of: a) quick and accurate regulation of muscular contraction; b) Co-ordination of the movements of the individual muscles; c) Precision of movement; d) Concentration; and e) Visual Control (Grandjean, 1988).

In this section, we present some factors arising from the human body that can impact the quality of music performance.

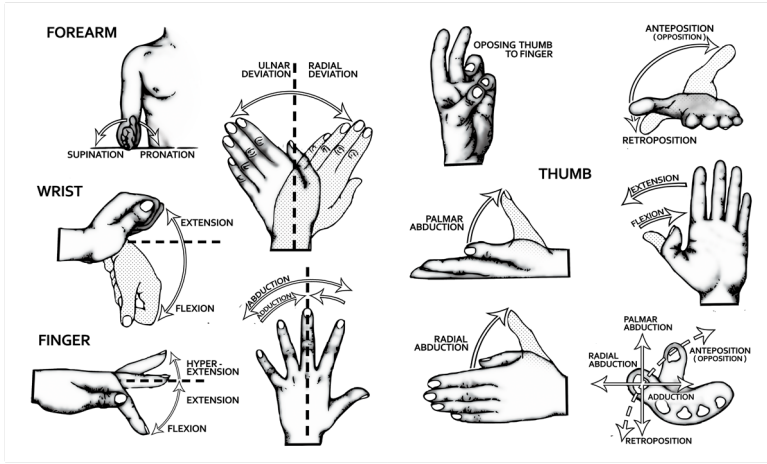
An Overview of Skilled Tasks

Skilled work is mostly a matter for the hands and fingers only (Grandjean, 1988). Hand movements are caused by muscles pulling on tendons to produce rotation around joints. Muscles in the forearm contract to rotate the hand around the wrist joint, while finger movements can be produced either by muscles in the forearm (extrinsic to the hand) or by intrinsic muscles in the hand itself (Wing *et al.*, 1996, p.35).

Fortunately, it is not necessary to be familiar with the complex anatomical structure of the hand to understand its movements. Hand function can be understood not just in terms of the anatomy and physiology of the upper extremity but also in terms of the computations that allow hands to manoeuvre as they do, named degree of freedom (*df*). With its 27 bones and 39 muscles, the hand has over 25 degrees of freedom (Wing *et al.*, 1996, p.170).

Figure 14 illustrates some of the hand movements and the terminology used to refer to them in the bibliography and throughout this work.

Figure 14: Terminology for hand movements.



Source: Redesigned image from the original available at ASSH - http://www.assh.org/Public/HandAnatomy/Pages/default.aspx.

When the number of muscles exceeds the number of the degrees of freedom provided by the joints, the system is said to be a biomechanically over-specified (Hogan, 1985). In other words, this means that the brain uses variable combinations of muscle activity to perform the same hand movement (Wing *et al.*, 1996) and although the movement is the same, the performance of the movement is not. Hence, body posture, limb orientation, and joint angles have a direct influence in the muscle length which is a determinant to speed and strength of the movement. For example, the strength in a knee extensor muscle is about 5-10 times greater with the knee flexed at 60-70º compared to the knee near full extension (about 10º of flexion) body position and joint angles (Kumar, 2004, p.46).

In order to perform a task, a skilled person has to use the most appropriate and effective combination of muscles, which is only possible by mastering this highly complex system. This requires continuous practice.

When a new technique (movement) is being learned, people often freeze their joints in an attempt to reduce the number of *degrees of freedom (dfs)* that need to be controlled; practice will allow them to free up these joints and capitalise on the interplay between them (Wing *et al.*, 1996). Furthermore, repetition leads to a gradual elimination of all muscular activity that is not essential to the skilled work, decreasing energy consumption and making the movement more efficient.

According to Carlevaro (1984), the gradual acquisition of mechanical ability or technique needs to be linked to various stages of development during a specific period of guitar training. At first, the different elements are studied in isolation; in a more advanced stage, they are grouped to form the proper technique. From each position and movement, every other is born.

Movement is a continuous flow of information that is not fully stored in the human brain. Findings from memory-for-movement

tasks have shown that people are poor at remembering movements but are good at remembering positions (Wing *et al.*, 1996, p.179).

The suggestion that the joints are frozen to facilitate the learning of a new 'movement' indicates that positions are used as markers to guide movements. The brain, however, does not store all possible positions. Instead, it stores a few postures and derives new postures from these representations (Wing *et al.*, 1996, p.179).

It has been demonstrated by Iberall (1989) that people typically constrain their handgrip configurations to a small number of patterns. These patterns are retrieved from the memory and applied to a whichever task resembles the scenario they originate from. If it is necessary, minor adjustments can be applied based on sensorial feedback. In the context of guitar performance, a bad posture can harm the guitarist and, consequently, his music. If a guitarist adopts a defective physical posture incompatible with his anatomy to perform then his technique and musical expression will be compromised (Carlevaro, 1984, p.3) Some of the postures and movements that expert musicians have to perform to play a musical instrument properly are unnatural and the classical guitar ranks among the most demanding instruments in this respect (Heijink and Meulenbroek, 2002). The problem is so severe, that 48-66% of string players report injuries serious enough to interfere with their ability to perform (Shan and Visentin, 2003).

In the next section, we describe some of the techniques, movements, and posture expected from a guitarist in order to play the guitar.

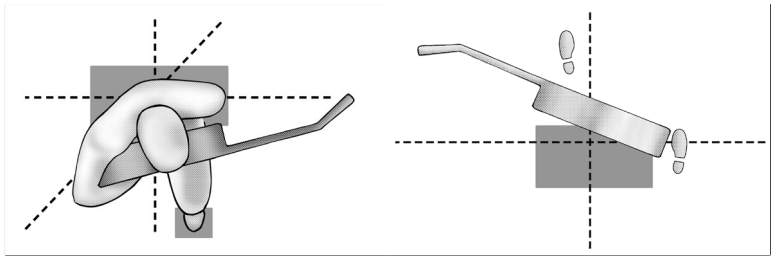
The Biomechanics of the Classical Guitar

This section intends to introduce an overview of the basic postures and movements used in the performance of the classical guitar. Although this study is not limited to classical guitar modelling, it is the classical guitar style that provides the most refined techniques and it is the classical guitar literature that best formally describes and explain the techniques.

The main reference for the topics discussed in this section is the work of Abel Carlevaro (1984), a virtuoso guitar player and teacher with a strong view on the biomechanical aspects of the classical guitar.

The starting point of what Carlevaro (1984) would consider a correct technique is related to holding and balancing of the guitar. A guitar wrongly placed, or a defective attitude in the manner of sitting will immediately generate difficulty in the action of the fingers.

Figure 15: Classical guitar holding position.



Source: Redesigned image inspired by Carlevaro, 1984, p.4.

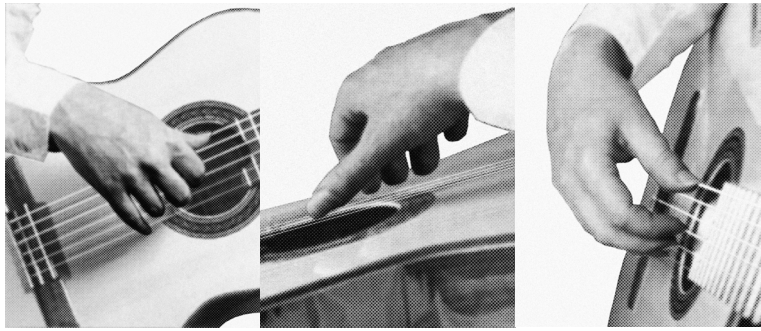
The guitar leans on the left leg (foot resting on a stool), the bottom side in contact with the right side of the body, the right-arm resting over the upper side of the guitar and the guitar neck pointing outwards (Figure 15). If the position is correct the guitar is stabilised by four active points of contact: (a) the left-leg, (b) right-leg, (c) the right arm, (d) the left hand. Three out of these four, need to be in contact all the time for the guitar to be stabilised.

Before we present a more detailed view of the way the left and right upper limbs work in the context of a guitar performance it is important to highlight that this book refers to right-handed guitarists as default. Normally, the right hand is used to pluck the strings whilst the left-hand stop the strings in the fretboard region. Nevertheless, some left-handed guitarists invert the strings and the hand's roles, but not all do. In fact, the traditional classical school of guitar maintains that left-handed guitarists must be taught in the same way as right-handed guitarists.

The playing area for the right-hand is considered to be the space between the bridge and the end of the fingerboard. The sound varies according to the exact point of contact: it is sharp and trebly close to the bridge and becomes progressively more mellow as you move towards the middle of the string (Denyer, 1992, p.73).

In a three-dimensional analysis of arm kinematics in violin performance, Shan (Shan and Visentin, 2003) demonstrated that the right-arm presents a higher motion than the left-arm. In a guitar performance, this is not the case. The right-arm works in a relatively immobile position serving as a point of contact to hold the instrument, as shown in Figure 16. The shoulder and elbow usually present a much lower amount of motion if compared to the wrist and fingers, which are the most stressed articulations of the system.

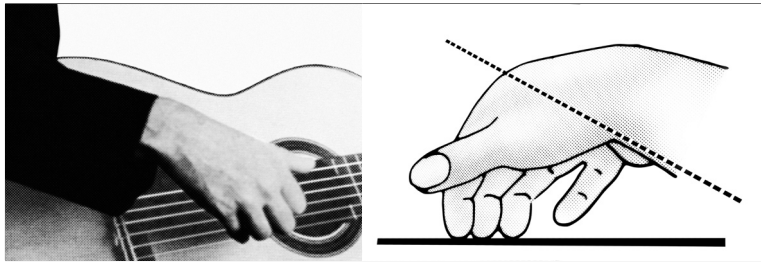
Figure 16: Classical guitar right-arm positioning.



Source: Redesigned image inspired by Chapman, 1994, p.60.

The natural positioning of the right arm is not completely compatible to the arrangement of the strings, so a small wrist flexion and radial deviation are required to align the fingers perpendicularly to the strings and make sure the thumb trajectory will not interfere with the index finger (Figure 17).

Figure 17: Flexion and rotation of the wrist (right-hand) in classical guitar.



Source: Redesigned image inspired by Carlevaro, 1984, p.12.

Biomechanically, the use of the right arm as a point of contact is not a good practice and may even harm the performance. If the guitarist applies excessive pressure with the internal side of the forearm, in an attempt to gain extra grip to stabilize the guitar then he also increases the internal friction of his tendons which slows down the finger flexing and extension.

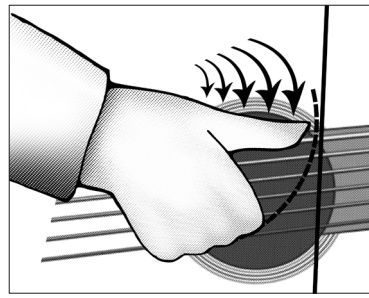
Another incorrect approach used to compensate a defective posture is to lean the right-hand against the bridge forcing the annular (ring) finger to bend and restrict its mobility. In essence, if the posture is not correct then compensation actions are likely to add stress to the overall system, ultimately compromising the performance.

Different musical styles and different guitar characteristics demand different right-hand techniques. While classical and flamenco guitars are played with the fingers, steel-string guitars are normally played with a plectrum (Chapman, 1994). The use of a plectrum generates greater stress in the extrinsic muscles of the hand because the wrist, instead of the finger, performs the movement. In this work, we will discuss only the finger-style technique.

In the fingerstyle technique, the thumb performs radial adduction/abduction and anteposition movements (Figure 14) to play laterally on the basses (Figure 18) and, exceptionally, on other strings too. This movement is carried out mainly by the intrinsic muscles of the hand, although the extrinsic muscles come into play when more power

is required. In the next section, we will discuss muscle properties, but for now, it is important to emphasize that these muscles work independently from muscles responsible for the flexion and extension of the other digits.

Figure 18: Right-hand thumb plucking movement in classical guitar.



Source: Redesigned image inspired by Carlevaro, 1984, p.29.

Note in Figure 18 that for optimum results the thumb must attack the string perpendicularly in a downward movement. Another angle would force the finger to slide on the string producing unwanted noises (Carlevaro, 1984, p.30).

The index and middle fingers must perform freely without hampering or being hampered by the thumb, usually in alternate upstrokes. The flexion and extension of these fingers are performed exclusively by the extrinsic muscle of the hands. The ring finger performs a similar movement to the index and middle finger but usually in the bottom strings (higher pitch - soprano voice). Anatomically, the ring finger shares the same muscles as the index and middle finger however in the mid-forearm, the main flexor muscle (FDP) divides into two: the radial and the ulnar. The radial part goes into the index finger, while the ulnar part goes into middle, ring, and little fingers. Consequently, the latter three fingers tend to move together, while the index finger can function independently of the others.

Playing a musical instrument, like the guitar, is a particularly skilful bimanual motor performance (Wing *et al.*, 1996). Both the

upper limbs are equally important in guitar performance modelling. This work investigates exclusively the role of the left upper limb in a guitar performance.

The Left Arm and the Fretting Hand

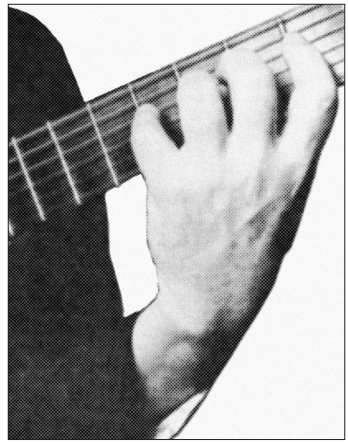
In our work, the left-hand is the main focus of the investigation. The decision of focusing on the left-hand does not imply it is more important than the right-hand in guitar performance. We prioritized the investigation of left-hand because Heijink and Meulenbroek's (2002) biomechanical study of the classical guitar has also focused on the left-hand allowing us a direct comparison of the results. Furthermore, in a bimanual activity such as the guitar playing, if both hands are investigated then the synchronism between them must also be investigated; therefore this is outside the scope of this book.

It is through a multi-finger coordinated motor task that the guitarist is able to press several strings at the same time in order to play chords. However, the fingers of the left-hand rarely perform in isolation; rather, they form a single unit with the wrist, elbow and shoulder joints to maximise their reach. Unlike the right upper limb, the whole biomechanical system of the left-arm is constantly changing to perform the left-hand techniques required in the classical guitar.

Carlevaro (1984) defines 'hand presentations' as the manner in which the fingers are located in relation to the fingerboard, which he classifies into: longitudinal, transversal, and combined (mixed) presentations.

In the longitudinal presentation (Figure 19), the fingers are positioned alongside the strings; one finger per fret. To perform this type of presentation, the left shoulder needs to drop allowing the elbow to come closer to the body. The forearm rotates to an almost full supination position allowing the wrist to remain flexed around 45º in a neutral (slightly radial) deviation. The fingers are arched to stop the string perpendicularly on the fingerboard (tip-pinch grip) avoiding any interference with the adjacent strings.

Figure 19: Longitudinal presentation of the left-hand in classical guitar.



Source: Redesigned image inspired by Carlevaro, 1984, p.65.

In the low frets region, the fingers in their most natural attitude cover no more than three frets. The same range in the higher region will naturally coincide with the four frets required in the classical guitar technique. This finger span of four frets means that the fingers will be positioned within approximately 2 cm of each other (abduction/adduction movement of the fingers). For anatomical reasons, the space between the middle and ring fingers will be generally smaller than the index-middle and ring-little fingers, making the performance of chords that demand the placement of the ring finger far from the middle ringer more difficult.

The flexion of the index, middle, ring, and little fingers involve the same group of muscles as the right-hand. The thumb, however, performs a different movement from that required to pluck the string (right-hand). To embrace the guitar neck the thumb of the left-hand performs a palmar abduction (Figure 20), which demands more power from the extrinsic muscle of the hands. This, however, does not mean that thumb must be vigorously pressed against the

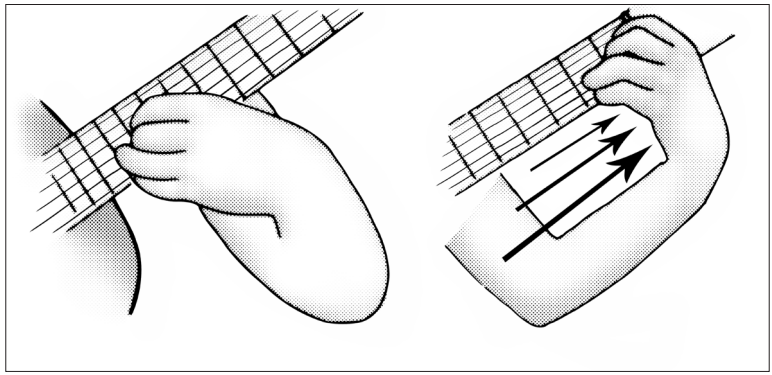
neck, just the opposite; it acts as a support to the other fingers and not the main force for grip.

Figure 20: Left-hand thumb positioning in classical guitar.

Source: Reprocessed image originally available at Tom Hess - http://tomhess.net/files/images/FAQ/Left_hand_2.jpg.

On the transversal representation, two or more fingers are placed in the same fret demanding an almost full radial rotation and flexion of the wrist to accommodate the fingers on the fretboard. Consequently, the elbow moves away from the body (Figure 21).

Figure 21: Radial deviation of the left wrist in at transversal hand presentation in classical guitar.



Source: Redesigned image originally from Carlevaro, 1984, p.66

As illustrated in Figure 21, the level of wrist flexion in this presentation is intimately related to presentation fret location. The further the hand moves away from the body, the more the wrist folds inwards (radial deviation).

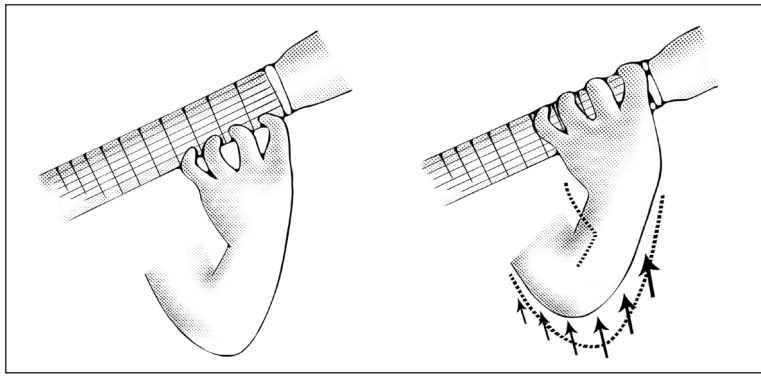
In practice, most of the presentations will be a combination of the longitudinal and transversal presentations, having one of two fingers presented in one way while the remaining fingers are presented in another (Figure 22).

Figure 22: Example of a combined presentation of the left-hand in classical guitar.

Source: Redrawn image originally from Carlevaro, 1984, p.66.

While the elbow and wrists provide the rotational movements that allow the guitarist to adopt the different hand presentations, the shoulder comes to action when the hand is required to perform a vertical displacement in order to reach the bass strings. It is important to highlight that the muscles of the shoulder are bigger and consequently slower than the extrinsic muscle of hand, therefore the force output is not as smooth, which generates jumps in the force delivery. In theory, this translates to slower and less precise movements. Figure 23 illustrates the mechanics of the movement.

Figure 23: Vertical displacement of the left-hand in classical guitar.



Source: Redrawn image inspired by Carlevaro, 1984, p.29.

The amount of force used to stop a note should be kept to the minimum, just enough to produce a clean sound. Excessive force detracts from freedom and agility of the movement of the finger, induces fatigue and can even produce an exaggerated hardening of the fingertip, diminishing the sensitivity and, consequently, the tactile feedback. In contrast, too little force will not stop the string appropriately generating a buzzed or muffed sound.

The Physiology of Guitar Performance

So far, we have seen the physical tasks that are required from a guitarist to perform. In this section, we will talk about the mechanism that carries out the task: the muscle.

Muscle Strength (Force)

Muscle strength is defined as the force a muscle can exert as a function of its contractile conditions, where contractile conditions depend on: a) the length of the muscle; b) instantaneous speed of shortening; and c) the history of length change (Kumar, 2004, p.45).

There are two kinds of muscular effort: a) dynamic - alternation of contraction and extension; and b) static - prolonged state of contraction (Grandjean, 1988). To cope with the different types of effort, our body produces three different types of muscle fibres:

1. Muscle Fibre Type I (SO – Slow Twitch): This type of fibre produces low force, it has an aerobic metabolism (uses oxygen to generate fuel - APT), it withstands long-term effort and it is found in small motor units dedicated to precise control of movement. It is very efficient and does not produce lactic acid and is, therefore, less vulnerable to fatigue;
2. Muscle Fibre Type IIb (FG - Fast Twitch): This type of fibre is found in larger motor units. Because they use anaerobic metabolism, they are much better at generating short (and fast) bursts of strength than muscle fibre type I. Their disadvantage is that they are more vulnerable to fatigue.
3. Type IIa (FOG): Known as intermediate fast-twitch fibres, these fibres use both aerobic and anaerobic metabolism almost equally to create energy.

Typically, a muscle will have all three types of fibres but in varying quantities. To some degree, the characteristics of the fibres can be modified through specific training although, to a large degree, the distribution of fibres is genetically determined (Freivalds, 2004, p.69).

Muscular adaptation involves thickening the muscle fibres and thereby increasing the total power of the muscle. Training for very rapid movement means not only increasing muscle power but also reducing internal friction by getting rid of some of the non-contractile material, such as connective tissue or fat. Therefore, it is appropriate to say that during the period in which a skilled operation is being learned, we can distinguish between two distinctive processes: a) learning the movements, and b) the adaptation of the organs involved in the task.

Normally, a musical performance is not a task that demands a high level of force. In fact, in many skilled activities, the skilled man is relaxed and economical in his movements, whereas the novice's work is cramped and tiring (Grandjean, 1988). Parlitz et al.(1998) have shown that amateur pianists not only use more force on every stroke but also used it for longer.

Even though the intrinsic muscles of the hand are predominately Type I (slow twitch), hand strength is not an issue for most skilled operations, such as the fine manipulation of a musical instrument. It is preferable to have less powerful but more controllable muscle in the hand than heavily loaded muscles (fibre type II) that are more difficult to control and to coordinate with others.

As previously explained, Type I fibres are good for static efforts but the guitar playing is mostly a dynamic type of effort. This incompatibility is solved in two ways: a) modifying the fibres distribution of the muscle through specific training; b) making use of a more suitable type of muscle through the development of a more refined performance technique.

Fortunately, the extrinsic muscles of the hand are responsible for flexing the fingers and they are predominantly composed by fibres of type II/IIa. These muscles are best suited for providing a continuous output force, whereas the intrinsic muscles, composed by fibre type I, act as stabilisers to metacarpal and phalange joints counteracting rotational movements (Chao, 1989). Nevertheless, the intrinsic muscles do play a role in the generation of small and finely graded forces of around 10% maximal voluntary contraction (MVC).

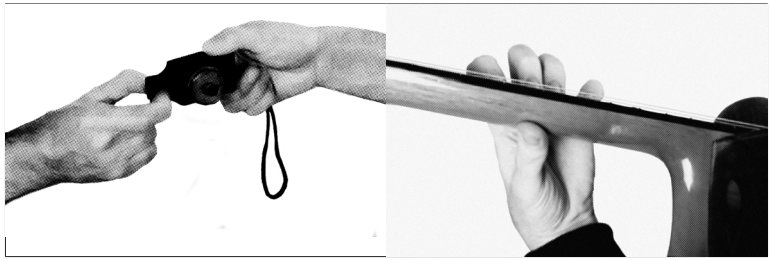
Below there are a few other pertinent remarks related to the use of the intrinsic muscles of the hand to generate force during a guitar performance:

1. Intrinsic muscles of the hand, rather than the extrinsic, are responsible for flexing or extending only one finger at a time. In a guitar performance this will happen then just one finger

needs to be moved while the others should remain in their positions (e.g. performance of grace notes);

2. Intrinsic muscles produce more force during the tip pinch than during power-grip (Wing *et al.*, 1996, p.84). The power grip is the one adopted when holding a tennis ball, for example. The tip-pinch is the grip that most resembles a finger position in guitar performance (Figure 24). This means that the intrinsic muscle of the hand may have to generate more force for playing the guitar than to perform any other task.

Figure 24: Tip-pinch grips illustration in classical guitar performance.



Source: Redrawn images originally available in Mathiowetz et al., 1985 (left) and http://www.acguitar.com/media_files/articles/193/23824/classic_left.jpg (right).

Although our mechanical analysis of the guitar has proved the opposite, Carlevaro (1984) believes that in a guitar performance there is no reason for a finger to apply more pressure than another. Either way, it is important to establish the maximum pressure (force) a finger is capable of producing.

Each muscle fibre contracts with a certain force and the strength of the whole muscle are the sum of these muscle fibres. The maximum strength of a human muscle lies between 0.3 and 0.4 N/mm^2 per the cross-section (PCSA); thus, a muscle with a cross-sectional area of 100 mm^2 can support a weight of 3-4 kg (30-40 N) (Grandjean, 1988).

The main force producer muscles in the flexion of the index, middle, ring are the Flexor Digitorum Superficialis (FDS) and the Flexor Digitorum Profundus (FDP). In a pinch grip action, the tendon forces were found to be in the range of 25 to 125 N (12.74 kgf) for the FDP and 10 to 75N (7.6 kgf) for the FDS (Freivalds, 2004, p.215). This power output is more than enough to produce clean notes on guitar as seen previously in Section 3.1.6, p. 56 – the measured force to generate clean notes on a real guitar ranged from 0.204 to 0.897 kgf.

However, the normal pinch grip only involves two fingers: the index and the thumb. These to fingers flex and extend using independent muscles, unlike in multi-pinch type of grip involving the index, middle and ring finger. As previously mentioned, the FDP muscle is shared by the index, middle, ring and little fingers. In a multi-finger pinch grip scenario, how much force would each finger produce?

In a power grip, it is known that individual fingers do not contribute equally to force production. The middle finger is the strongest at 28.7% of the grip force, followed by the index, ring, and little fingers, with percentage contributions of 26.5, 24.6, and 20.2% respectively (Freivalds, 2004). A muti-pinch grip is not a conventional type of grip so the force distribution values could not be found in the literature.

Nonetheless, it is important to emphasise that in guitar performance the strength of the fingers does not come exclusively from hand intrinsic and extrinsic muscles, but a from a muscular system in which the hand is just one participant (Carlevaro, 1984, p.172). However, the maximum strength application is limited by the weakest segment or joint implicated in that particular activity, which is usually the intrinsic hand structure.

We conclude this section reporting some facts related to force production that could affect a guitar performance:

1. The optimal operating range of a muscle is in the middle of the articulation working range (plateau region), as verified Heijink and Meulenbroek (2002);

2. Female grip strength typically ranges from 50 to 67% of male grip strength but 35% of the gender difference can be explained by hand size (Kumar, 2004, p.183);
3. Non-preferred handgrip strength is 80% of the preferred handgrip strength(Shock, 1962); A '10 % rule' (dominant hand is 10% stronger) it is also often referred in the biomechanical literature (Schmidt and Toews, 1970);
4. With the increase of age, muscles become smaller. Consequently, the strength and speed of the movements decrease. This decrease appears to occur more significantly in people over the age of sixty (Kumar, 2004, p.75);

Endurance and Fatigue

Endurance is defined as the ability to persist in a physical task and is typically measured in time (Kumar, 2004, p.85). In other words, it is the ability to sustain continuous dynamic contraction or isometric contraction for a prolonged period of time. Duration, intensity and frequency must be considered when analysing endurance.

Endurance and fatigue are reciprocal concepts however they are not the same. Fatigue prevents the continuation of a physical task that reaches the limit of endurance; two people can have the same fatigue and yet vary in endurance.

Fatigue denotes a loss of efficiency and a disinclination for any kind of effort, but it is not a single, definite state. In physiology, muscular fatigue is a phenomenon that reduces the performance of a muscle after stress, not only reducing its power but also slowing the movement (Grandjean, 1988, p.175).

According to Carlevaro (1984, p.22), the isolated work of the fingers is the main cause of muscular fatigue in guitar performance. This occurs because the *Type IIa* muscles involved in the flexion of the fingers (FDP and FDS) are overly stressed by a poor technique.

Although daily training can lead to a gradual muscular adaptation by thickening the muscle fibres, it can not reduce muscular fatigue if the real cause of this fatigue lies in defective technique (Carlevaro, 1984, p.23). In a correct guitar technique, the effort should be distributed within the muscles of the hand, wrist, and arm.

A professional guitarist is, supposedly, someone who is highly aware of the correct guitar technique and very fit for the task. Even so, he will eventually succumb to fatigue. When this happens, not only will his muscular ability to perform the task is compromised but also his perception of time, making the task appear longer that it actually is.

To exemplify the level of effort that can be encountered in guitar performance, consider an F major chord executed as shown in Figure 25. If we consider the Antoria guitar and calculate all the measured force (Figure 10) required for producing a clean note in all the positions of this chord, we would end up with the cumulative force of 2.28 kgf to perform this chord shape in the first fret. The same chord shape in the 10th fret would require a force of 3.52 kgf to be performed.

Figure 25: F (barre) chord shape.

Fonte: Source: Creative Commons by Mjchael is licensed with CC-BY-SA-3.0 (https://commons.wikimedia.org/wiki/File:Chord_F.jpg).

Normally, an additional 10-40% of extra-force is unintentionally used as a safety margin (Wing *et al.*, 1996), which would increase the force to 4.92 kgf.

A barre-chord is a type of palmar pinch grip. The average maximum force of a palmar-pinch grip of the left-hand of male adults stays around 10.4 kgf (Mathiowetz *et al.*, 1985), hence a 47.3% MVC is required to perform this chord.

Danion and Galléa (2004) propose that steady force output by the fingers can only be maintained at a level 30-40% MCV, meaning that anything above this range can only be maintained for a short time usually bellow 6 seconds. Carlevaro (1984, p.104) himself proposed exercises capable of draining the muscle in 15-20 seconds.

Muscles that are not directly involved in the force production undergo fatigue too. If the primary muscles suffer fatigue, an unintended contraction of other muscles induces a change in posture to alleviate the primary fatigued muscles. This means that the task will be performed by muscles that are not the most effective to the task, inducing loss of precision and increasing the risk of errors. This phenomenon is known as *contralateral* activation and it is more evident in highly repetitive tasks or tasks that require awkward postures, such as guitar playing.

Fortunately, because of the redundancy found in the hand's biomechanical system, changing the force application along the finger axis provides an important variation in the participation of muscles in force exertion allowing for the temporary relaxation of the fatigued motor units (Kumar, 2004, p.185). Therefore, with the correct technique, it is possible to manage some situations that could lead to fatigue.

Speed

The speed of the muscle contraction is a powerful determinant of muscle strength and, as a consequence, the speed at which a physical

work task is performed. For instance, the shortening of a muscle at a mere 10% of its maximal speed causes approximately 50% loss of strength from the isometric force (Kumar, 2004, p.46).

While muscle force is proportional to physiological cross-section area (PSCA), muscle speed (or excursion) is proportional to fibre length. Muscles of different architecture, but the same fibre type, may differ in strength and speed by factors of ten or twenty (Wing *et al.*, 1996, p.69).

The maximal velocity of the fibre shortening (V_0) is expressed in terms of contractile element lengths. Wickiewicz, Roy, Powell and Edgerton (1983) suggests the maximum speed for the contraction of human muscles is about 8 lengths/s for slow-twitch and 14 lengths/s for fast-twitch. These values, however, need to be treated with some reservation because they were extrapolated from mixed fibre muscles experiments.

The fibre type composition of the finger flexor muscles (FDP and FDS) is mixed, with a slightly lower proportion of type I fibres (Maurer *et al.*, 1995; Mizuno, 1994). Considering the FDP (index finger) fibre length of 61 mm (for FDS is 31 mm), a full contraction would take around 70 to 125 ms.

As presented in Section 3.2.3.1(p. 74), a full contraction of the muscle would produce more force than is necessary to play a clean note, which is on average around 0.423 kgf. The question is how fast the flexor muscles can produce just sufficient force to play a clean note. This force-velocity relationship has attracted much interest in muscle physiology since this relationship contains useful information concerning the basic mechanisms of muscle contraction (Wohlfart and Edman, 1994).

Hill (1938) was the first to demonstrate that the force-velocity relationship in skeletal muscle can be adequately described as part of a rectangular hyperbola. This force-velocity curve for a contracting muscle can be written as:

Equation 4: Hill Equation – Force-velocity of contraction muscles.

$$(F + a)(v + b) = (F_0 + a)b$$

Where *F* is the force generated by the muscle, *v* is the velocity of shortening, F_0 is the maximal isometric force at optimal contractile element length, and *a* and *b* are constants with units of force and velocity, respectively (Kumar, 2004, p.62).

> 'a/F_0' and 'b/V_0' are dimensionless quantities of an approximate value of 0.25 for many muscles across species and temperatures (Hill, 1938), including human fast-twitch fibres at 37°C (Faulkner *et al.*, 1986; Faulkner *et al.*, 1980).

Measuring the maximal velocity of shortening in human skeletal muscles is difficult (Kumar, 2004, p.64); however, a rough estimative may be obtained by determining the maximal power output of a group of muscles as a function of movement speed. Knowing that the maximal power is achieved at 31%, it is assumed that a/F0=b/v0=0.25 (Herzog, 1994). Power is given by $P = Fv$.

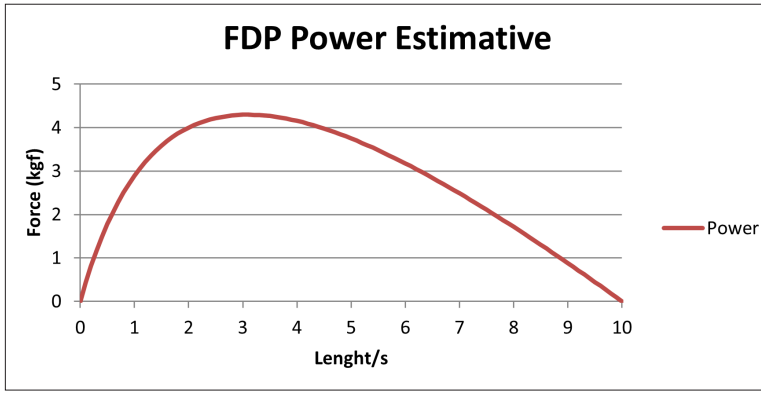
Re-arranging the Hill's equation, we can calculate Force as:

$$F = \frac{(F_0 + a)b}{(v + b)} - a$$

To exemplify, let us suppose the FDP is the only force producer for the flexion of the index finger. The FDP has cross-sectional area (PSCA) of 177 mm^2 (Doyle *et al.*, 2003, p.107), meaning its maximal isometric force $F_0 = 177 \times 0.3N/mm^2 = 53.1N/mm^2$ *(5.41 kgf).*

Because the FDP is a mixed composition fibre type with slightly less proportion Type I, we will consider its V_0 = 10 lengths /s. The constants are given by: $a = F_0 \times 0.25 = 1.125N/mm^2$; and $b = V_{0\,x}\ 0.25 = 2.5$ *length/s.*

Figure 26: FDP muscle estimated output power.
The x coordinates represent the contraction rate measured in length per second and the y coordinates represent the output power measured in kilogram-force.



Source: Own image.

Figure 26 shows the estimate output curve for the FDP, where around 0.2 of its fibre length the muscle produces 0.81 kgf, enough to play a clean note in virtually any position of the fretboard. In a time measuring unit, 0.2 length/s corresponds to 20 ms.

Other Factors that Impact Musical Performances

Training level: Skilled performance declines with the retention interval although the rate of forgetting can be slow, sometimes non--existent. Different skill types have different lengths of skill retention. Perceptual motor skills, such as playing an instrument, driving, flight control, and most sport skills, generally demonstrate that very little is forgotten for long periods of time. In contrast, procedural skills, which require a sequence of steps, such as how to use a text processor are more rapidly forgotten (Wickens and Hollands, 2000, p.283).

Music Performance Anxiety (MPA): MPA or stage-fright is a serious problem that has prevented many excellent musicians

from pursuing a career in music. This is caused by fear of failure that often becomes a source of distraction and leads to poor performance (Thompson et al., 2006). There is a suggestion that MPA could also have a profound effect on the efficiency of the movement of muscles in fingers.

Temperature: Skin temperature has a profound effect on nerve conduction parameters. The effect on latency caused by temperature drops may be as large as 0.1 ms/° and increases almost linearly by 2.4 ms/°, as the temperature increases from 29 to 38°C (Johnson and Olsen, 1960). The protocol to take measurements of muscle strength and speed demands requires a body temperature of 37° C.

SUMMARY

Errors in music performance have been studied mainly in the domain of cognition, which Palmer and Van de Sande (1993) referred to as production errors. On the physical level, mistaken actions are studied in the Ergonomics field but the focus is on avoiding errors rather than modelling them in order to recreate them.

Imperfections in guitar playing technique often give rise to discrepancies between the intended music and what is actually produced (Carlevaro, 1984, p.92); these errors can be called slips. In Section 3.1 we have explored some of the mechanical aspects involved in playing the guitar and how some 'unwanted' noises are produced, discussed at 3.1.6. We have seen that forces between 0.204 and 0.897 kgf are necessary to produce clean notes on a guitar.

In section 3.2 we have presented how the guitarists' body copes with the task of guitar playing and some of the biomechanical and physiological reasons that could prevent him to operate the guitar as it should to produce a clean performance.

These reasons are related to: force, speed, and precision. Each of them relates to particular errors in guitar music performance. For example, lack of force can lead to muffling of buzzed notes.

We have also explained how fatigue leads to precision errors. If the primary muscles normally used to perform a task are fatigued then secondary muscles are recruited. However, the secondary muscles might not be fast, precise, or powerful enough for the task.

A common problem found is the recruitment of highly loaded muscles to perform skilful tasks. If the output power provided by these muscles is not smooth (fibre type II) than skilled tasks that require precision are more difficult to perform, consequently, more errors are produced. Quite intuitively, faster movements terminate less accurately, whereas targets covering small areas requiring increased accuracy are reached with slower movements (Wickens and Hollands, 2000, p.13). Therefore, it is fair to presume that neighbouring areas (adjacent string and frets) are more likely to be hit by accident.

In order to verify the importance of the speed, force and precision of guitarists in a music performance we have designed a set of experiments to measure these parameters. These experiments are discussed in the next Chapter.

Chapter 4

Guitar Performance Data Acquisition and Analysis

> Music has just as much to do with movement and body as it do to soul and intellect. (Esa-Pekka Salonen)

In the field of ergonomics, performance measurements are generally associated with one of four categories: measures of speed or time, measures of accuracy or error, measures of workload or capacity demand (how difficult is to use the product), and measures of preference (Wickens and Hollands, 2000, p.13). In biomechanics, performance is mostly characterised in terms of endurance, strength, speed, and accuracy (Sanders and McCormick, 1993, p.215).

Even though we acknowledge the relevance of all these attributes in guitar performance, we limit ourselves by measuring only the attributes that we believed would have the greatest impact on the outcome of guitar performance computer modelling. These are: precision/accuracy, speed, strength/force, and posture.

The tasks involved in the biomechanical musical experiments must be very simple in both cognitive and musical terms (Heijink and

Meulenbroek, 2002); By choosing simple tasks it is possible to minimise or completely exclude delays originating from the cognitive difficulties. Wargo (1967) estimates that this delay ranges from 113 to 528 ms, but it can be reduced with training and anticipation of the movement.

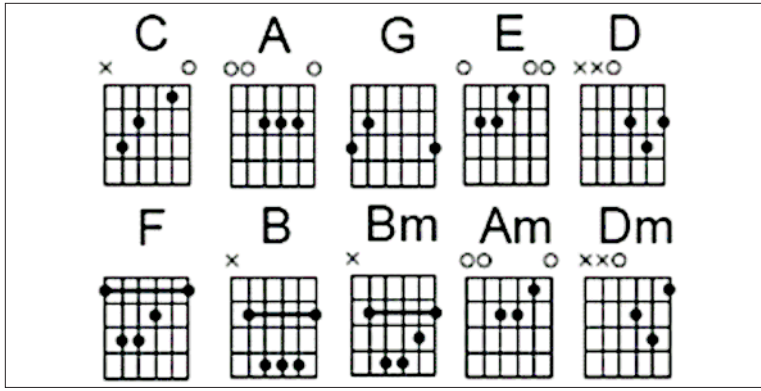
In biomechanical terms, the performance of a monophonic line is simpler than a polyphonic line because it does not necessarily require the coordination of multi-fingers in a chord execution. Based on that, Heijink and Meulenbroek (2002) opted to use monophonic phrases (scales) in their biomechanical study of guitar performance

However, most finger movements made by primates (including humans) are not isolated movements of a single digit (Wing *et al.*, 1996, p.81) as earlier observed by Parncutt (1997) in his ergonomic studies of pianists. The reason for the involuntary movement of adjacent fingers is that the FDP muscle located in the midarm (one of the main source of power for finger flexion) divides into two parts: the radial and the ulnar. The radial part goes into the index finger, while the ulnar part goes into middle, ring, and little fingers. Consequently, the latter three fingers tend to move together, while the index finger can function independently of the others (Freivalds, 2004). As a result, we have decided, rather than following Heijink and Meulenbroek's (2002) monophonic strategy, to make use of chords in our experiment.

The chords we have selected are often taught in the first stages of guitar training. This increases the chances that the guitarists will be familiar with the selected chords and, as a consequence, reduce the cognitive activity involved in the performance of these same chords (coordinated reflex).

In musical terms, chords are a group of notes played together. In biomechanical terms, this means that the guitarist is likely to have to use more than one finger to be able to perform the chord. But, because the guitar is a polyphonic instrument, there are different ways to perform the same chord. We will refer to the way the chord is performed as a chord shape. Figure 27 shows the chord shapes of the chords used in the experiments.

Figure 27: Chord shapes used in the experiment.
The 6x4 matrix objects seen in the image represent a guitar fretboard; the black circles inside the matrix a show the positioning for the fingers. The horizontal line linking two points represent a 'barre'. The hollow circle on top of the matrix indicates an open string and the 'x' mark indicates the string should not be played.



Source: Own image.

We have decided to adopt a system known as *CAGED* and *EDAm* to select the 'shape' (fingering) of the chords (Edwards, 1983). With the addition of a barre[2], the chord shape that composes this system becomes 'moveable'. For instance, Chord B (with the barre addition) has the same basic shape as the chord A but two frets closer to the guitar's body.

The selection of well-known chord shapes that have been learned in the early stage of guitar training contributes to the extrapolation of the results to other chords with similar chord shapes. This is possible because the brain works with similarity meaning that instead of storing all possible body positions, the brain derives new postures from a few basic ones (Wing *et al.*, 1996). Three male right-handed

2 The barre technique is used to stop more than one string using just of finger, normally the index finger.

guitarists, aged between 19 to 30 years old, took part in the experiments. All of them have had at least two years of classical training but just one considers himself a classical guitarist. The others sought specialisation in more popular genres such as jazz, rock and blues. The subjects have between 6 and 20 years of guitar playing experience.

Despite the qualitative characteristics of our research, we have tried to keep the factors that could possibly add uncertainty steady, facilitating a comparison between the subjects. As previously seen in Chapter 3, some of these factors are: gender, age, body (skin) temperature, hand dexterity, state of training, muscle constitution, momentary motivation, and fatigue (Grandjean, 1988).

In the biomechanical study of Heijink and Meulenbroek (2002), six male professional classical guitarists ageing from 22 to 36 years old were assessed. In comparison, a sample of three guitarists does not appear to be large enough to represent a population and, in fact, it is not. However, rather than seeking the generalisation of a population, we were pursuing an understanding of an individual guitarist.

In summary, more time was invested and more data was collected from every individual subject rather than focusing on the group as a whole. To put this into perspective, if we compare only the first experiment against the whole experiment of Heijink and Meulenbroek, our subjects played an average of 540 notes against 264 notes played on Heijink and Meulenbroek's (2002) experiment, an increase of 104%.

SHARED PROTOCOL FOR BOTH EXPERIMENTS

To ensure the optimum performance of the subjects a strict protocol was put in place before the start of the measurements.

1. The methodology and objectives of the experiment were explained to the subjects as well as their right to withdraw at any time. If the subject wished to proceed, the authorisation form was signed. The subjects were paid £6 per hour for their participation.

2. To avoid electromagnetic interference in the recording equipment, mobile phones or any other unnecessary electronic equipment were switched off and/or removed from the room;
3. The subjects were asked whether they had any injury or abnormal circumstance that could impact his ability to play the guitar;
4. The subjects were instructed not to play any stringed instrument within the 12 hours previous to the experiment. This cautious recommendation was made to avoid accumulated fatigue. However, due to the nature of the muscles used in the task and the level of MVC[3] expected, a full recovery should take no more than a few minutes;
5. In terms of work design, it is inadvisable to perform precise activities immediately after heavy work since some smaller motor units may be fatigued and larger ones will be recruited with less precise control (Freivalds, 2004, p.69). If requested, a fifteen-minute rest was provided to attenuate any fatigue of small motor units that could have occurred during commuting to the lab (e.g. cycling);
6. The body temperature was taken and ensured to be around 37ºC;
7. A reasonable amount of time (relative to the experiment) was given to the subjects to familiarise themselves with equipment. This was done through proposed finger warm-up exercises. Cold fingers lose their sense of touch and motility and become numb (Singleton 1972, p.42)
8. Once the experiment had finished, the subjects were debriefed.

3 Maximal Voluntary Contraction.

EXPERIMENT 1: SPEED AND PRECISION

Music performance is a skilled activity that demands very fast and precise movements from the performer. Thompson and Dalla Bella (2006) have shown that pianists may be required to play up to 30 sequential notes per seconds over extended musical passages.

As an athlete, a 'virtuoso' instrumentalist is the result of years of exhaustive training in which his body and mind go under continuous adaptation to maximize his genetic pre-disposition to the task. A performer is always faced with two problems. One concerns the purely mechanical difficulties contained within a musical work, the other the interpretative and expressive aspects of the music. It is highly advisable to tackle the latter problem first, for the artistic domain should be entered from the very beginning (Carlevaro, 1984).

Abel Carvelaro, a virtuoso guitarist himself, believes that the precision and efficiency of the movement is tied to the mental representation that prepares the mechanisms used to perform the action. He says: "One of the precious faculties of a true performer is knowing how to select his movements" (Carlevaro, 1984, p.21).

According to Wickens (2003), one of the ways to improve the speed of the movements is to anticipate them, reducing the number of possible alternatives when the time to act comes. A less obvious strategy is to use body members closer to the cortex to reduce neural transmission times that could vary from 100 m/s to 25 m/s, respective to the larger and smaller (more precise) type of neurons found in the Central Nervous System.

> The secret of remarkable manipulative skills of the human lies in the way manual tasks are organised and controlled by the nervous system says. (Johansson, 1996, p.381)

Even though the subjects were given enough time to practise every single chord shape before the readings were taken, the selection

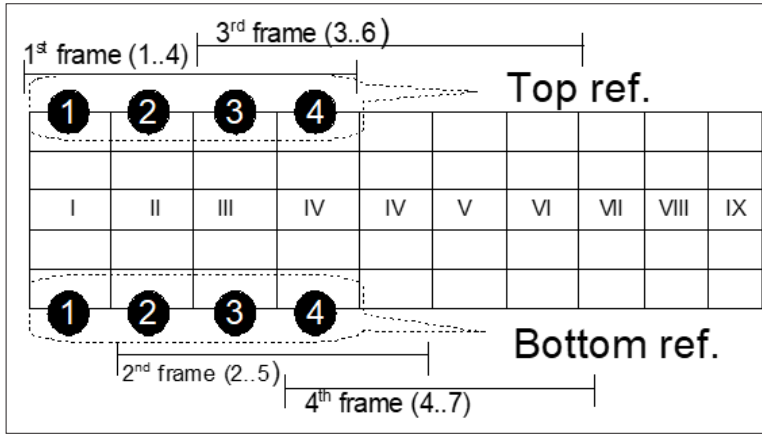
of trivial chord shapes enables the neurological process of movement anticipation. This suggests that, if no other motor control constraint is considered, the performance time required for performing the chord shapes should be very similar. This refutes the hypothesis that some chord shapes takes longer to perform than others due to biomechanical constraints.

Rosenbaum (1996) has already proven that motions can be made more rapidly in certain ways and directions because of the nature of human physical structures. This theory was formalised in a travel-cost function for motor behaviour. As previously explained, his theory assumes the use of basic 'stored' postures to create new ones; the selection of which 'stored' postures to use is based on the effort to move from one posture to the other. This concept has been widely used in computational fingering models for guitar.

Of course, a travel function implies a departure and arrival point or posture. To minimize the interference regard to the initial hand's position prior to the chord shape performance we have established frames of reference. A frame of reference is a term that came from classical mechanics where the validity of Newton's laws of motion were associated with an inertial reference point, for example, one that was fixed to the earth (Wing *et al.*, 1996).

In our experiment, two frames of reference were proposed: one in the top (6^{th}) and another in the bottom (1^{st}) string of the guitar (Figure 28).

Figure 28: Frames of Reference proposed for the experiments.
The horizontal lines represent the guitar strings; the vertical lines represent the frets. The black circles with a numeral inside indicate the positioning for the fingers, where 1 = index, 2 = middle, 3 = ring, 4 = little finger.



Source: Own image.

So far, I have described how time and distance have been considered in this experiment but there is another variable closely related to these two: accuracy. Fitts (1954) was one of the first researchers to look into this multi-variable correlation proposing an equation which later became known as Fitt's law. According to Fitt's law, faster movements are less accurate, whereas precise moments are slower (Wickens and Hollands, 2000, p.387). This reciprocity between time and errors has been well documented across different areas and constitute one of the fundamental tenets of ergonomics, referred to as the index of difficulty of the movement (Wickens *et al.*, 2003, p.263).

In our experiment, we hoped to demonstrate this positive correlation between response time and error rate by requesting the subjects to perform the chords as fast (and accurately) as they possibly can.

Although precision errors were not in the scope of Heijink and Meulenbroek (2002) experiments, they did acknowledge that the

positioning of the finger too far to the left would lead to a buzzed note and too far to the right a damped (muffled) note. On average, they found a variance in the spatial domain in the range of 4 mm in a guitar with inter-fret spacing varying from 12.9 to 36.5 mm.

Further, Heijink and Meulenbroek (2002) revealed other interesting findings; the index finger was placed farthest from the fret as opposed to the little finger that was placed closest. The middle and ring ringer were placed in between these two. The authors suggested that the player must have positioned the index finger farthest from the fret to avoid damping the string whilst the little finger was placed closer once it is almost perpendicular to the fretboard.

Another strategy to avoid performance error is the progressive positioning of the fingers closer to the frets in higher frets (toward the guitar body) given that, due to the mechanics of the guitar, in the low end (closer to the head) the fingers can be placed slightly further from the frets without compromising the sound quality of the note.

Finally, they reported that hand repositioning and finger span also affect precision. The fingers were placed closer to the frets when hand positioning was not required and when the fret span was small.

The total time of the experiment, including explanations and repetitions whenever applicable, was around two hours for which the subjects were paid £12 each.

Measuring System

A theory is formulated based on the results obtained through logical reasoning, observations and/or experiments (Dodig-Crnkovic, 2002). The use of a scientific methodology requires that the results must be reproducible, in the sense that people do not have to believe the presented data or results; rather they should be able to repeat the experiment and validate the results by themselves. The selection of a cost-effective apparatus used in the data collecting is critical to this matter.

One of the most popular pieces of equipment used to track movements (and speed) is a 3D motion tracking system, such as the Optotrak 3020 (NothernDigital, 2009). Heijink and Meulenbroek (2002), Bejjani and Halpern (1989), Shan and Visentin (2003) are some of the researchers that have opted to use such type of device in musical performance biomechanical studies. The system is very precise (0.2 mm in each dimension) but also costly, costing over $60,000.

Although cost is a concern, it is not the only one. In order to pick up the movement variations, the system requires that markers, in the form of infrared-light-emitting diodes (IREDs), need to be strapped on the subject's joints (Figure 29). This might not be a problem to a runner or golfer, but the interference of these IRED can cause in such delicate operations as musical performance is a concern.

Figure 29: Infrared-light-emitting diodes (IREDs) used in tracking systems. Markers used in conjunction to Optotrak 3020 3D motion Tracking System.

Source: Redrawn image originally available in NOTHERNDIGITAL, 2009.

In the Heijink and Meulenbroek (2002) experiment, five IREDs were taped on the nails of the guitarist, one in the back of the left--hand, two in the proximal phalange on the index and middle fingers of the right hand and three on the body of the guitar. No discomfort was reported but judging by the manufacturer's advertising image (NothernDigital, 2009), illustrated by Figure 29, it is fair to assume that the device, as big as a fingernail, was likely to interfere in the guitarist's abilities to perform. As a result, we have opted to use a less intrusive and cheaper option: a guitar-like MIDI controller Yamaha EZ-AG shown in Figure 30.

Figure 30: Yamaha E-AZ Guitar.

Source: Redrawn image originally available at Yamaha - http://www.yamaha.com.

The Yamaha EZ-AG simulates the dimensions of electric guitar however instead of strings the controller has buttons on the fretboard. When pressed, these buttons trigger MIDI messages that are sent to the Sound Generation Unit.

We developed a bespoke real-time MIDI recorder to interpret these messages and record not only the speed but also any (precision) error occurred during the experiment. Most importantly, this piece of software was able to validate the data informing us in real-time whether the experiment had to be repeated. Once the experiment

was finished, the data was saved as a Microsoft Excel compatible file for further analysis.

Task

The subjects were instructed to perform the chord shapes shown in Figure 27 throughout the entire extension of the fretboard within frames of reference that go from first up to the ninth fret (Figure 28).

The use of frames of reference in different regions of the fretboard serves two purposes. Firstly, it establishes a common ground for comparison between the subjects by setting an initial posture for reference. Secondly, it will allow us to understand the influence that the initial hand position has in the overall time taken to perform the transition.

As Figure 28 illustrates, the references were set both at the bottom and top 'strings' of the instrument. The index, middle, ring and little fingers of the left-hand had to press and hold their respective reference positions within a four frets' range in which the chord was going to be performed. The speed was calculated based on the time difference from the moment all four reference buttons were released up to the moment all notes (buttons) of the chord shape were depressed. No right-hand action was required or recorded.

Before the experiment started, the subjects were given a scale exercise to warm up their fingers and to get used to the instrument. Only when they declared themselves comfortable and apt to perform the tasks did the recording start. This phase took around 5 to 10 minutes.

The experimenter then asked the subject to set the bottom reference at the frame [1..4] and 'jump' to the first chord shape as fast and as precisely as he possibly could using a previously agreed fingering. The procedure was repeated until frame [9..12] was reached, for all 10 chord shapes, from the bottom, and top references. The chord shape recording order was: C, A, G, E, D, Am, Dm, F, B, and Bm.

If the subjects felt the need they were given a few extra minutes to practice the task before the actual recording took place. At the end of every recording, a preliminary analysis of the data was done to validate the readings. In case of any undesired abnormality, the experiment was repeated.

The experiment was paused every 30 minutes for a 10-minute break or whenever the subject reported fatigue.

Data Analysis

The subjects had to perform ten chord shapes (three times each) in nine different regions of the guitar, departing from two points of reference. Even if no mistakes were made, over 10,000 messages were recorded for every subject. Unfortunately, the MIDI data transfer is serial and highly subject to latency (up to 7 ms detected in our setup) so the messages had to be sorted and the latency removed before analysis, which was done by the custom-made MIDI recorder.

The time taken to perform the chord shape was calculated by subtracting the timestamp of the latest button pressed (representing a note of the chord) from the timestamp of the earliest reference button release.

The data were recorded continuously per chord shape. If for any reason the reference is not set properly or not all notes of the chords are performed then segmentation of the data into individual trials would be compromised and, as a result, the speed could not be calculated. To overcome this problem, we developed a simple algorithm to identify a common pattern of errors on the data, like a note addition or deletion.

The algorithm implements a segmented search on the data that goes beyond the immediate instance of the pattern that is being sought. For instance, consider a string sequence 'B123B234456B456'. The character 'B' marks the beginning of the three digits numeric sequence (e.g. '123'). The first instance is a perfectly formed 'B123'; the second instance is 'B23445', which has an extra '456'. The algorithm does not stop and proceeds with the search, finding another perfectly formed instance

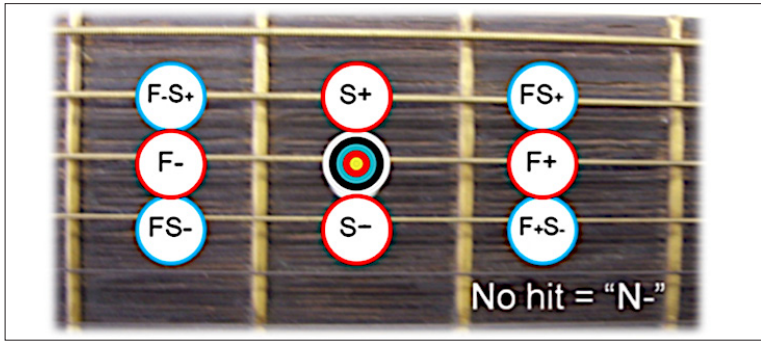
'B456'. The slot with the error can then be isolated and analysed in detail. Is the '456' an error of the second instance or a mal-formed third instance? To answer that we need to identify what we were expecting to find, which in this case was the instance 'B456', hence a '456' is more likely to be a third instance lacking the reference 'B' than three additional digits of the second instance. This example demonstrates how it is possible not only the correct measurements of speed but also determine accuracy (errors) by identifying note addition and exclusions.

There were situations however when automatic identification and correction of the errors in the data was not possible. Since only three readings per chord and fret were taken, no automatic outlier removal strategy could be used. Instead, a manual selection of the value (usually the median) of each trial was used to calculate the average speed of the chord shape per fret. The chord shape speed per subject is calculated as the average time taken to perform the chord shape in all frets.

The overall speed of the chord shape was calculated as the mean of the chord shape speed for all three subjects. The same rationale was used to calculate the speed of the individual fingers.

In order to quantify the accuracy of the subjects, we have classified the errors using a dart target-like system, as seen in Figure 31.

Figure 31: Error coding system.
'S' =String, 'F' = Fret, '+' = Above or Right, '-' = Bellow or Left.



Source: Own image.

In this target-like strategy of classification, every error receives a code indicating the distance from the target. In the code system [S] stands for string, [F] for fret, '+' for top or right-hand side, and '-' for bottom or left-hand side. As an example, suppose that the target is the position [2, 3]. If the finger hits the positions [3, 3] and [2, 3] at the same time, this error is classified as 'S+'. If there is no hit for a particular position, then this error is classified as 'N-'. In the unlikely event of a hit outside the immediate peripheral area then a numeral is added (i.e. 'S+3').

Of course, this system is only possible if the fingering used by the performer is known beforehand. Table 6 shows the fingering adopted by the subjects.

Table 6: Subjects choice of fingering used in Experiment 1.

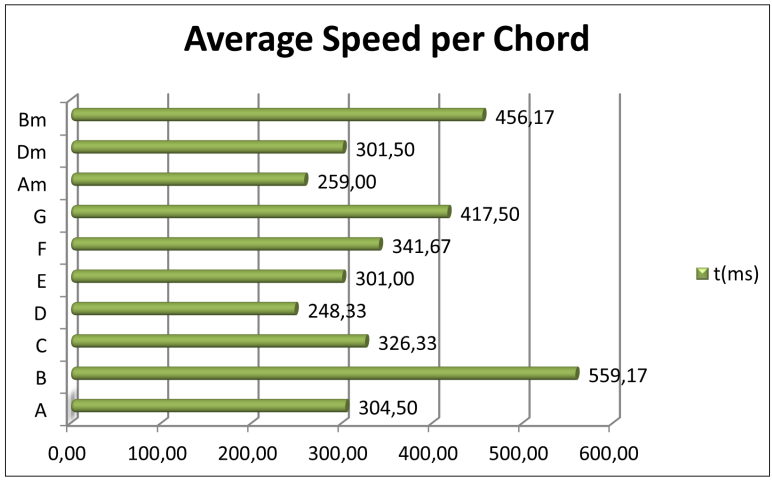
Chord	[string, fret]	Subject 1	Subject 2	Subject 3
C	[5,3]	Ring	Ring	Ring
	[4,2]	Middle	Middle	Middle
	[2,1]	Index	Index	Index
A	[4,2]	Index	Index	Index
	[3,2]	Middle	Middle	Middle
	[2,2]	Ring	Ring	Ring
G	[6,3]	Middle	Middle	Middle
	[5,2]	Index	Index	Index
	[1,3]	Little	**Ring**	Little
E	[5,2]	Middle	Middle	Middle
	[4,2]	Ring	Ring	Ring
	[3,1]	Index	Index	Index
D	[3,2]	Index	Index	Index
	[2,3]	Ring	Ring	Ring
	[1,3]	Middle	Middle	Middle
Dm	[3,2]	Middle	Middle	Middle
	[2,3]	Ring	Ring	Ring
	[1,1]	Index	Index	Index
Am	[4,2]	Middle	Middle	Middle
	[3,2]	Ring	Ring	Ring
	[2,1]	Index	Index	Index

Bm	[5,2]	Index	Index	Index
	[4,4]	Ring	Ring	Ring
	[3,4]	Little	Little	Little
	[2,2]	Middle	Middle	Middle
	[1,2]	Index	Index	Index
B	[5,2]	Index	Index	Index
	[4,4]	Middle	Middle	Middle
	[3,4]	Ring	Ring	Ring
	[2,4]	Little	Little	Little
	[1,2]	Index	Index	Index
F	[6,1]	Index	Index	Index
	[5,3]	Ring	Ring	Ring
	[4,3]	Middle	Middle	Middle
	[3,2]	Middle	Middle	Middle
	[2,1]	Index	Index	Index
	[1,1]	Index	Index	Index

Results

The average time for the subjects to perform a chord was around 350 ms. The D chord was the fastest at 248 ms and the B chord was the slowest taking more than twice as long at 559 ms.

Figure 32: Average speed to perform a chord.
The x-coordinates represent the time in milliseconds and the y-coordinates the chords measured.



Source: Own image.

There are some explanations for the slower performance of the B chords. Firstly, the palmar pinch used in the barre technique requires a different set of muscle that is stronger and slower. Besides, the B chord also requires an awkward upper-limb configuration in contrast to Am and E which are anatomically very comfortable to the subjects.

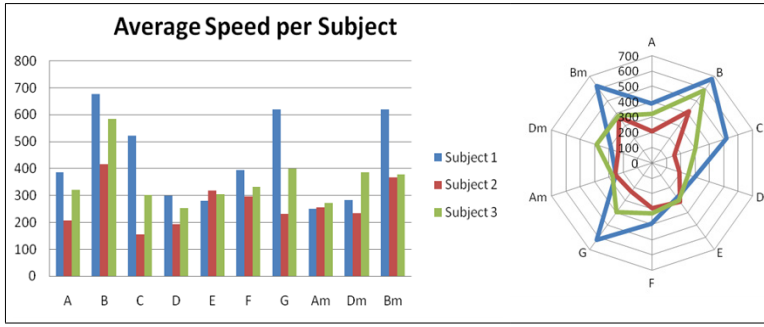
Secondly, the B chord shape requires a small hand-motion, from the first to the second fret of the frame of reference. The same applies to the Bm chord, which is the second slowest. Note that Heijink and Meulenbroek (2002) have also detected the influence of the hand repositioning in the speed.

Lastly, the number of digits involved in performing barre--chords is greater and as a consequence, the task is more complex. It must be remembered that the overall speed of the chord is equal to the speed of the slowest link (digit) of this system, which according to Freivalds (2004) is the litter finger.

Evidence suggesting the retardant effect caused by the use of the little finger can be found when analysing the fingering used by the Subjects to perform the G-chord (on average the slowest of the non-barre-chords). While subjects 1 and 3 used the little finger in the position (1, 3), Subject 2 preferred to use the ring finger instead (Table 6).

Figure 33 shows that, proportionally to the readings of the other chords of the same subject, the G Chord was performed much faster by Subject 2 than Subjects 1 and 3.

Figure 33: Average speed to perform a chord per subject.
In the barre chart (left) the x-coordinates represent the chords and the y-coordinates the time in milliseconds. The radar chart (right) allow another comparison highlighting the patterns of speed per chord between the subjects



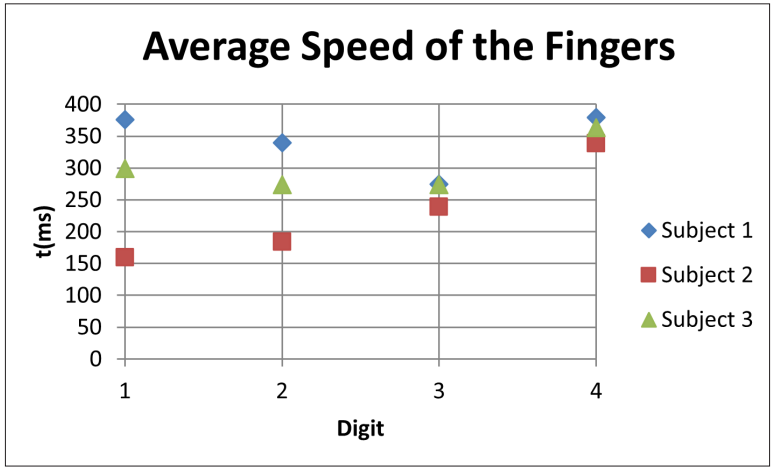
Source: Own image.

By comparing the average speed of the subjects per chord, it is possible to see that Subject 2 was consistently faster except for the chords of Am and E. The time variation of all the subjects to perform these two chords was very small, respectively 36 and 24 ms. Heijink and Meulenbroek (2002) reported variations in the time domain among their subjects around 25 ms.

Figure 34 shows the average speed per finger. As previously suspected, the little finger was indeed the slower one. Surprisingly,

the ring finger has shown similar values for all the subjects, being the fastest finger for Subjects 1 and 3.

Figure 34: Average speed of the fingers when performing the proposed chords. The x-coordinates represent the finger, where 1 = index, 2= middle, 3 = ring, and 4 = little; the y-coordinates show the time in milliseconds.

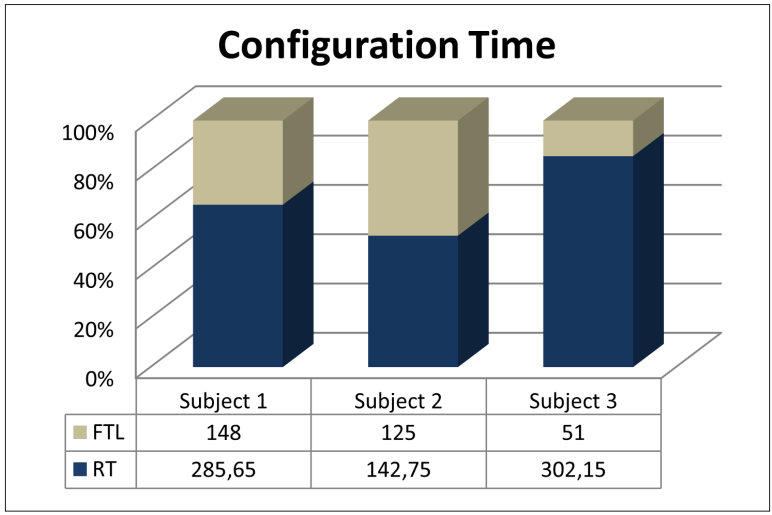


Source: Own image.

Through analysis of the speed of the digits, we could observe a pattern in the strategy of positioning the finger on the fretboard. While Subject 2 seems to have made constant use of the index finger as a guide, Subject 3 preferred to group his fingers before positioning them.

Figure 35: RT and FTL speeds.

The percentage shown in the y-axis is related to the subject average time to perform the chord shapes. FTL = First to Last and RT – Reaction time.



Source: Own image.

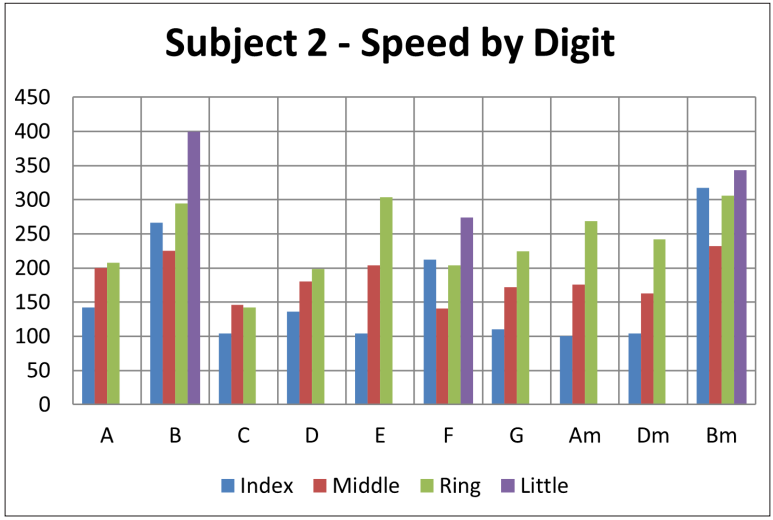
To help us understand these strategies, the overall time to perform a chord shape was decomposed into a) Reaction Time (RT): the time it takes to configure and move the hand to the region where chord shape must be performed; and b) First-To-Last note time interval (FTL): the time elapsed from the moment the first and last finger was actually put into place.

The FTL is an especially important measure because it helps to reveal trends in the use of the fingers. If the FTL time is small in comparison to the overall performance time then it suggests that the fingers are being grouped and then the buttons pressed together. Conversely, if the FTL time is high in comparison to RT then one finger may have been used as a guide to set a reference to the fingers.

The guide-finger strategy is something that the classical technique strongly recommends avoiding. Carlevaro, in 1984, already

considered the use of a guide finger obsolete (Carlevaro, 1984, p.79), but this still seems to be common practise among Jazz guitarists who adopt a less strict performance technique to match the interpretational freedom characteristic of the Jazz style. In this technique, the guide-finger searches for a note of the chord (usually the fundamental note) and only then are the rest of the fingers laid to form the chord. From the perspective of the motor control system, the use of a more precise digit as the guide-finger could help the performer to build an imaginary image of the fretboard in which the guide-finger sets a spatial reference for the other (less precise) digits as well as providing tactile feedback that later can be verified by the auditory or visual senses.

Figure 36: Subject 1 speed of the fingers per chord.
The x-coordinates represent the chords and the y-coordinates the time in milliseconds.



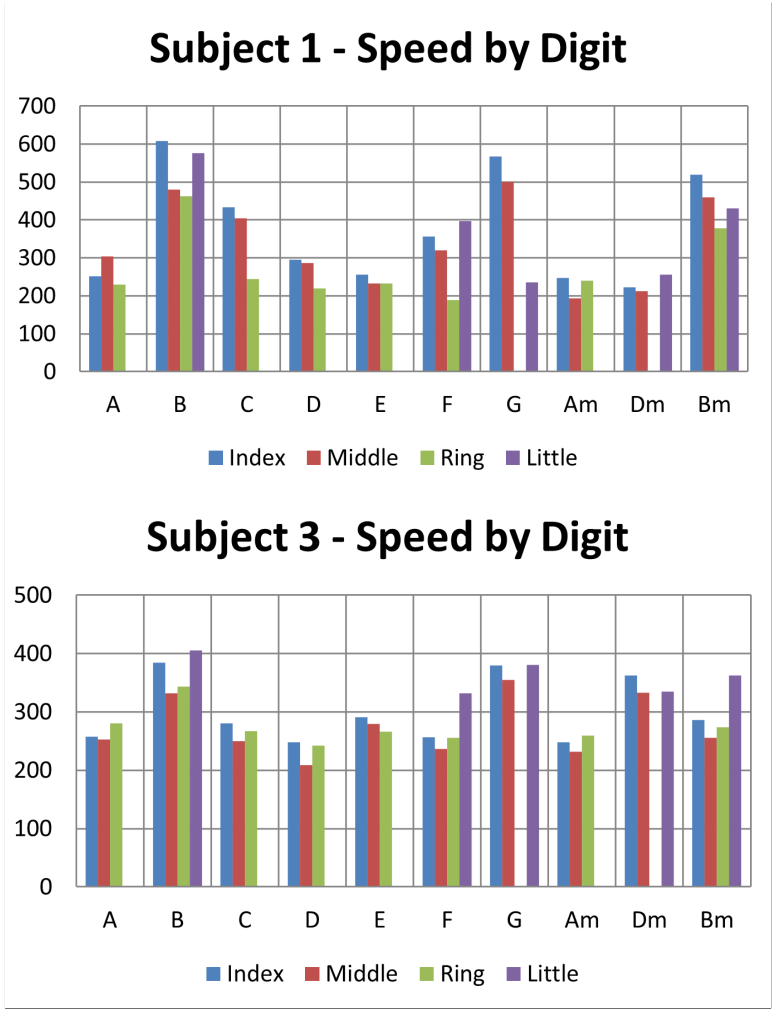
Source: Own image.

Figure 36 shows Subject 2 constantly placing the index finger firstly at all the non-barre chords. In the case of the barre-chords, the middle finger was placed first.

Using a radically different approach, we have Subject 3 (Figure 37) who has consistently positioned all the fingers on the fretboard in a very short time, a technique considered to be more refined. Contrast Subject 1, who has not adopted any easily recognisable pattern, suggesting a cruder technique.

Figure 37: Subjects 1 and 3 speed per digit.

The x-axis shows the chords and the y-axis the time in milliseconds.

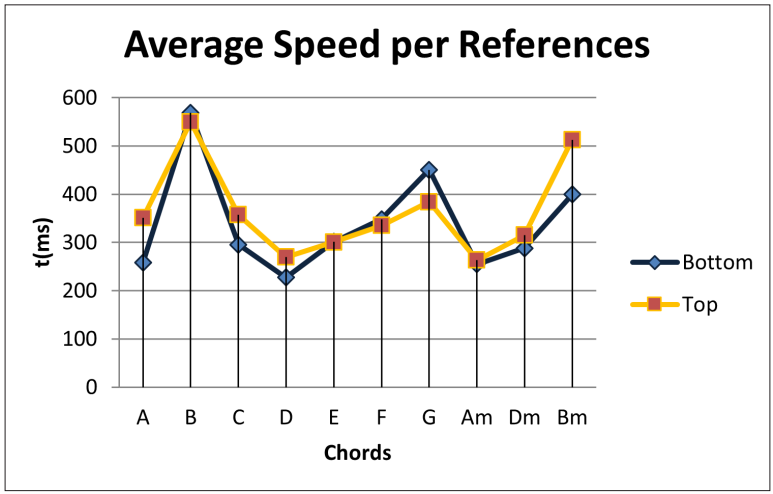


Source: Own image.

As previously explained, by setting the departure (references) and the destination (chord shape) points of the movement we have attempted to normalise its trajectory allowing a more direct

comparison between the subjects. The use of top and bottom references attempts to average out possible discrepancies in chords being performed closer to a reference than another. At this point, the deviation resulted from vertical displacement was not found to be statistically relevant as seen in Figure 38.

Figure 38: Top and Bottom speeds comparison.
The x-coordinates represent the chords and the y-coordinates the time in milliseconds.

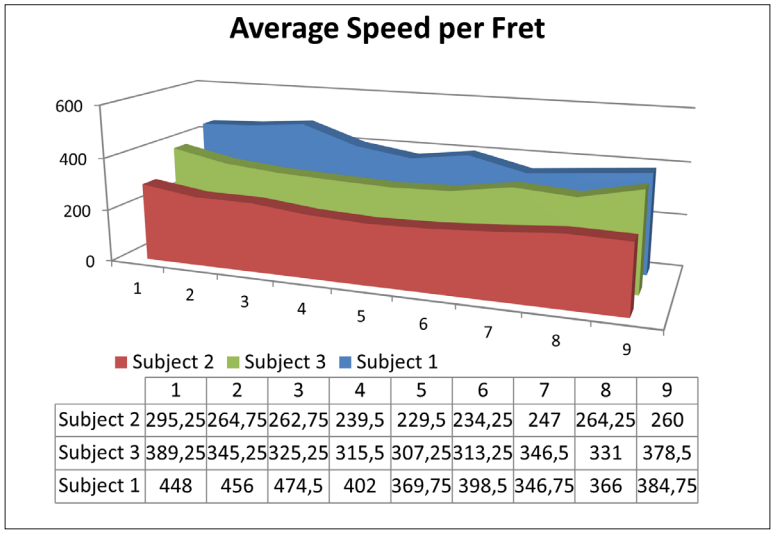


Source: Own image.

Heijink and Meulenbroek (2002) have found that guitarists prefer to keep their joints in the middle of their range when performing. This is due to the lower cost of this position. We requested the subjects to perform in different regions of the fretboard forcing them to perform out of the articulations' preferred range. Indeed, the subjects performed slightly faster towards the middle of the fretboard, where the elbow and shoulder joints operate in the middle of their range (guitar sitting on the right leg). The average speed per fret can be seen in Figure 39.

Figure 39: Average speed per fret.

The x-axis represents the fret region and the y-axis the time in milliseconds.

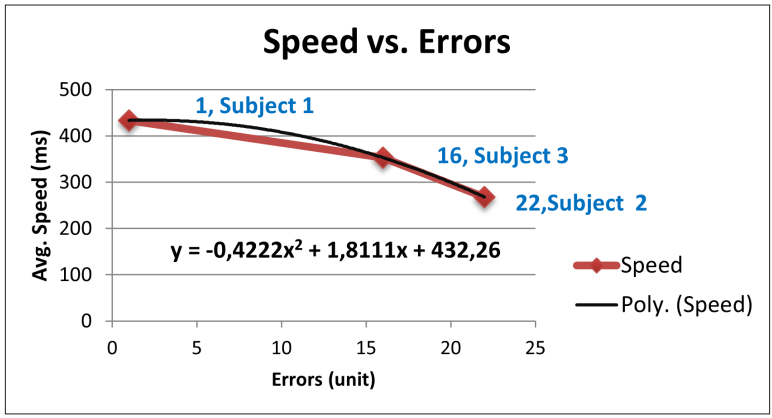


	1	2	3	4	5	6	7	8	9
Subject 2	295,25	264,75	262,75	239,5	229,5	234,25	247	264,25	260
Subject 3	389,25	345,25	325,25	315,5	307,25	313,25	346,5	331	378,5
Subject 1	448	456	474,5	402	369,75	398,5	346,75	366	384,75

Source: Own image.

Figure 40: Speed and Error correlation.

The x-axis represents the number of errors and the y-axis the time in milliseconds.



Source: Own image.

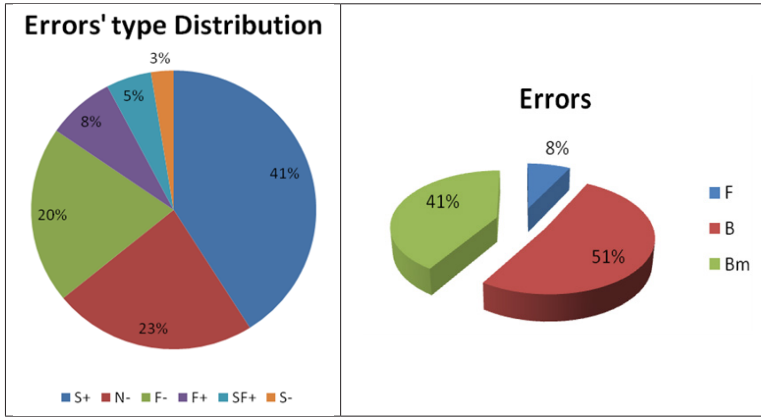
Although the experiments were not designed to prove the existence of a speed-error trade-off, we could find evidence that Fitt's law also applies to guitar performance. As can be observed in Figure 40, the fastest subject was also the least precise whilst the most precise was the slowest.

According to our results, the B chord was not only slower but it was also the most difficult to play. From the total errors, 51% were generated during the performance of the B chord, followed by Bm and F chords, with 41% and 8% respectively.

It is well established that acquiring barre techniques is a difficult stage in learning to play the guitar. The strings dig into the joints and the softer parts of the index finger causing discomfort (Chapman, 1994, p.78)

Discomfort, however, seems not to be the only factor that could lead to errors. The Yamaha EZ-AG guitar has buttons instead of strings and yet the errors only happened during the performance of the barre-chords (Figure 41).

Figure 41: Proportion of the recorded errors per type.
'S+' =hit string above the target, 'N-' = note missing, 'F-' = hit in fret left to the target, 'SF+' = hit string above and fret in the right to the target, 'S-' = hit string bellow the target.

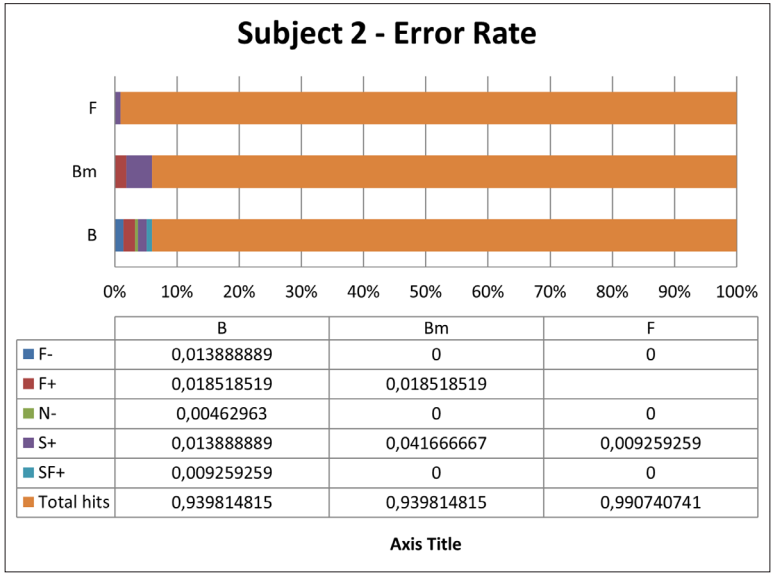


Source: Own image.

From total recorded errors, 41% were from the 'S+' type meaning the subjects hit a string above the target. Analysing this figure further we can find that 87.5% of these errors were on the Bm and B chords, both using a barre that covers from the 1st to 5th string.

If the subject applies a barre from the 1st to 6th string but does not pluck the 6th string, this will have little impact in a produced sonority. Some performers may not even consider it an error at all. For our system, however, this still counts as an 'S+' error. In reality, 78.5% of the 'S+' errors in these two chords were caused by the index finger, used in the barre technique.

Figure 42: Subject 2 probability error rate.
The y-axis shows the percentage of the type errors type per chord.



	B	Bm	F
F-	0,013888889	0	0
F+	0,018518519	0,018518519	
N-	0,00462963	0	0
S+	0,013888889	0,041666667	0,009259259
SF+	0,009259259	0	0
Total hits	0,939814815	0,939814815	0,990740741

Source: Own image.

Figure 42 shows the probability of Subject 2 incurring an error when performing the barre-chords. Note that Subject 2 recorded the highest number of errors. Bm and B have the same statistical

probability of performance errors but the repertoire of errors found in B chord is much more diverse.

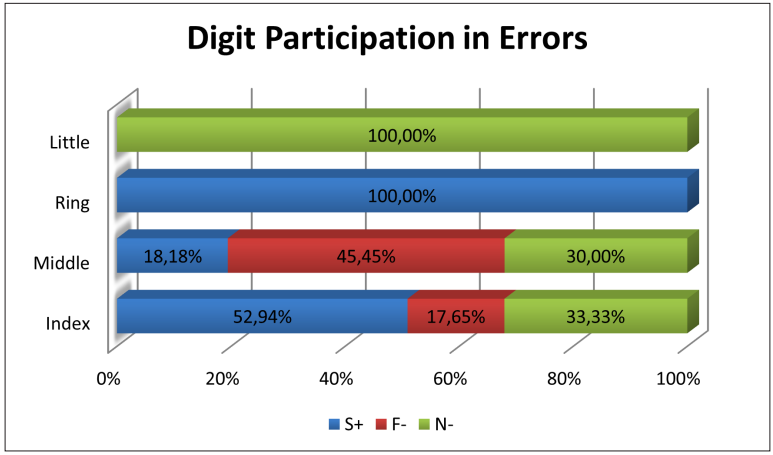
A profile of the error can be drawn based on its location and the finger used. Table 7 shows all the B chord errors for Subject 2. A simple analysis can reveal interesting patterns of error. For instance, all the 'F-' errors were caused by the index finger, departing from the bottom reference, and took place on the first string.

Table 7: Errors recorded from Subject 2.

Error type	Subject	Finger	String	Fret	Chord	Direction
F-	2	1	1	5	B	Bottom
F-	2	1	1	7	B	Bottom
F-	2	1	1	8	B	Bottom
F+	2	1	5	3	B	Bottom
N-	2	4	2	4	B	Top
S+	2	2	5	4	B	Top
S+	2	1	6	3	B	Top
S+	2	2	5	8	B	Top
SF+	2	1	6	3	B	Bottom
SF+	2	1	6	11	B	Bottom

Overall, the index finger was responsible for 43% of errors, followed by the middle, little and ring fingers with 28, 10, and 2% respectively. It is important to remember that these errors were related to barre-chords, therefore the index finger had the highest probability of erring, having to press 5 or 6 buttons at the same time.

Figure 43: Fingers participation on errors.
The x-axis shows the percentage of the finger's participation in the particular error types where S+' =hit string above the target, 'N-' = note missing, 'F-' = hit in fret left to the target.



Source: Own image.

Although the little and ring fingers have a smaller contribution to the total of errors, they were more consistent in a particular type of error, as seen in Figure 43. All the errors 'caused' by the little finger were from the 'N-' type, which one could assume is related to its lower strength if compared to the other digits.

This shows that errors can not be analysed merely by quantitative terms. In order to truthfully model precision errors, qualitative aspects of the error must also be considered.

EXPERIMENT 2: FORCE AND POSTURE

The amount of pressure exerted by the guitarist's finger on the strings to stop them against the fretboard could impact the quality of the note produced as explained in the previous chapters.

Even though Heijink and Meulenbroek (2002) recognised that the left-hand fingers must produce relatively large forces, they have not measured it. In fact, to the best of our knowledge, no specific study on the guitarist's strength has yet been published. So, instead of relying on the data produced by the anthropometric and biomechanics community, we had to develop our own methodology and devices to capture this data.

Even for experts, strength measurement poses some challenges; the scope of the experiment needs to be very well defined. As Kumar (Kumar, 2004, p.200) observed, any index of human performance presents a significant level of variation therefore standardisation in experimental protocols is needed.

One of the first proposals towards the standardisation of hand movements was the Jebsen Test of Hand Function (JTHF) (Jebsen *et al.*, 1969). The JTHF is a reference list of hand movements that are present in the Activities of Daily Living (ADL). As one could imagine, the movements involved in musical performance are not always as natural as those found in standard daily activities.

Another approach classifies the way the hand is shaped to manipulate objects, known as handgrips. These grips are very generic and rather simplistic if compared with the movements and postures required to perform on a guitar. Nevertheless, two of them are undoubtedly present in a guitar performance: a) palmar pinch grip used in barre-chords; b) tip pinch grip used to play a single note (not on the open string).

As the name suggests, the pinch grip is characterised by the junction (opposition movement) of the thumb with any other digit (pinching). In the context of guitar performance, this grip presents a small but crucial difference: the left-hand thumb uses the guitar's neck as an extension of itself to balance out the 'pinching' of the other digits, possibly all four of them and not just one as would be expected in a normal pinching grip. The presentation of the barre-chord will require an even more complex configuration of the hand, merging

the palmar pinch grip of the index finger with a multi-tip pinch grip of the other digits.

Strength can be measured under one of two conditions: dynamic conditions, when the body member is being moved (isotonic or isokinetic strength), and static conditions, when the force is applied against a fixed object, with no displacement of the body member (isometric strength) (Sanders and McCormick, 1993, p.216). For this experiment, the strength will be measured mostly under static conditions but we should not expect the subjects to remain immobile for the entire recording session.

The configuration of the upper extremity positioning plays a very significant role in the hand's strength capabilities (Kumar, 2004, p.200) so the posture of the subjects should be considered with great care. The American Society of Hand Therapists even recommends physically restraining some of the subject's irrelevant movements.

A less radical proposal was made by Mathiowetz (1985) who recommends that during hand tests the subject should be seated with the shoulder adducted and neutrally rotated, 90º elbow flexion, and forearm and wrist in neutral position (Mathiowetz *et al.*, 1985).

Naturally, limiting the subjects' movements is not a feasible option in our context. Although the classical guitar school establishes the 'correct' technique to perform the movements involved in guitar playing, there is a great deal of personal preference involved in these movements. As we previously explained, human manipulative skills come from an over specified biomechanical system that allows us to perform similar movements in different ways and we believe this variation should be measured.

The only option remaining is to also measure the subject's postures. Marshall (1999) argues that the joint deviation and limb posture measurements are as important as the force measurement itself.

In summary, to measure the isometric force exerted by the guitarist's left-hand fingers we need equipment capable of recording (a) multiple hand grips using in a (b) dynamic condition scenario, and

(c) that allow changes in body postures. Unfortunately, such a specialised device could not be found so we had to design and build our measuring device: a dumb guitar equipped with force sensors that are 'played' by the subjects while wearing a skeleton that records their movements, as detailed in the next section.

Measuring System

In the past, biomechanical/kinematical studies of musical performance often required the development of new devices and techniques that would allow the measuring of the biomechanical variables without disrupting or interfering in the performance itself. Carl Seashore (Seashore, 1936), one of the first to conduct psychological studies of music performance, developed a piano camera system to record gestural data from hammer and foot-pedal movements.

More recently, these studies have been carried out using a combination of audio analysis algorithms and motion tracking systems, as Heijink and Meulenbroek and others (Burns and Wanderley, 2006; Penn *et al.*, 1999; Shan and Visentin, 2003) used in their studies of violin and trumpet music performance.

As previously observed, the motion track system is very precise but the placement of the IRED requires special care to not limit the performer's movements. This limitation has been solved by Burns (2006) who developed a method to visually capture the fingering using multi-camera setup and a finger-tracking algorithm.

Although the camera-based system is very efficient for kinematical measurements, it can not measure force and this is where the audio analysis usually comes into play. Unfortunately, this is also not possible in a guitar performance because the amplitude of the sound is dictated by the right and not the left hand.

In our experiment we have used two independent devices to measure posture (movement) and strength, respectively a Gypsy6 exoskeleton and a custom made guitar-like device equipped with Tekscan

Flexiforce sensors A201-25 (0-25lbs/11.3kgf) that we have designed and built. Figure 44 shows a subject trying out the setup for the experiment.

Figure 44: Subject trying the force measuring apparatus.
Setup composed of a Gypsy6 exo-skeleton and FOGU – a specially made guitar that records the coordinate finger's force production.

Source: Own image.

The Animazoo Gypsy6 Torso motion capture system is a MIDI skeleton equipped with 18 sensors (16 potentiometers, 2 gyros) with a resolution of 0.125 degrees each, positioned on the wrist, elbow and shoulder joints (Animazoo, 2009). A dedicated computer running a real-time MIDI recorder (the same as the one used in the speed experiment) saves all the data for further analysis. It is an available commercial product.

Finding an appropriate device to measure the left-hand strength in a guitar performance is not an easy task. Penn (1999) said that large

muscle groups can be studied using isokinetic fatigue protocols but an isokinetic device to test small hand muscles is still lacking, therefore a more creative approach remains necessary. To investigate the first dorsal interosseous muscle in pianists, Penn (1999) used surface electromyography (EMG) but the focus of his study was not specific to force, but fatigue.

Another possible approach is to equip a real instrument, whenever possible, with thin-film force sensors as Parlitz (1998) has done in a force-related piano experiment. This film, however, needs replacing after every trial which is not practical.

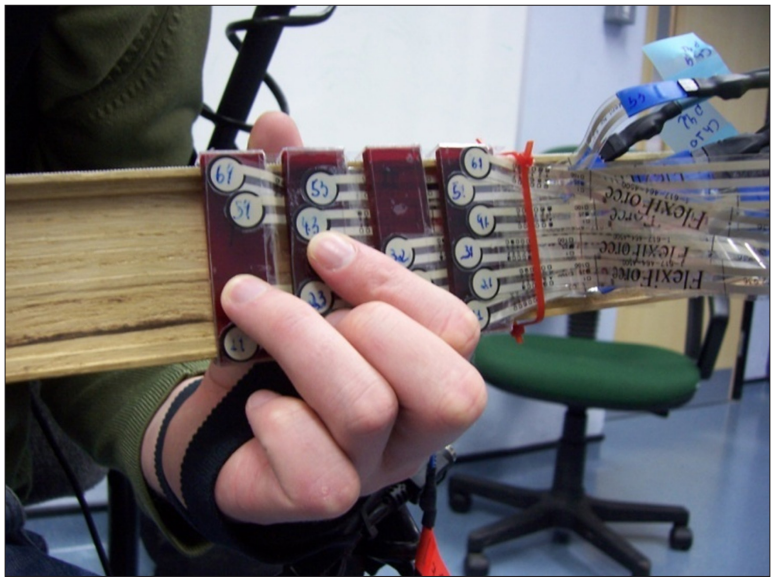
We designed and built the device used to measure strength; we called it FoGu (Force Gauge Guitar).

FoGu is a guitar-like device with similar physical dimensions to a classical guitar, except the neck width, which is 58 mm against the traditional 52 mm. This increase in width is necessary to accommodate the force sensors and, although not ideal, it should not have a major effect in the force exerted by the hand considering the tasks do not involve large finger spans. Besides, a seven-stringed guitar and early baroque guitars use 58-60 mm neck width (Bennett and Dawe, 2001) so this dimension does not make the guitar impossible to play.

The force readings are registered by 18 sensors glued in four moveable plates that slide along the fretboard locking into pre-established positions equivalent to the inter-fret spacing of the classical guitar. The distribution of the sensors was optimised to use the fewer possible number of sensors to perform all ten chord shapes (Figure 27).

The dimensions of the plates were calculated based on 9th to 12th fret dimensions of a guitar with the same scale length, respectively: 20.5 mm, 19.5 mm, 19 mm, and 18.5 mm. In the lower frets, space was left between the plates to simulate the normal inter-fret spacing. The detail of this sensor distribution can be seen in Figure 45.

Figure 45: Close up view of distribution sensors in FoGu.



Source: Own image.

The electrical output from the sensors was sent to an analogue-to-digital converter - IRCAN Ethersense Interface - through a series of individual circuits built using 100k resistors to maximise its sensitivity. The Ethersense interface converts the analogical signal to digital and sends it to a Max/MSP patch. This patch generates an OSC (Open Sound Control) message that is captured by a custom-built Max/MSP patch (Figure 46). The custom-built patch analysed the input, mapped the sensors' output to the chord shapes/fingers (Table 8) and then recorded the force per finger.

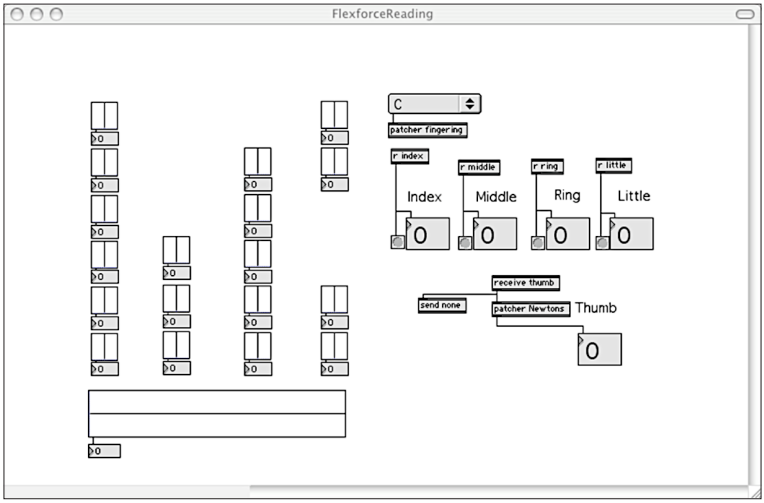
Table 8: Sensors mapping to the chord shapes.

The settings used for the Ethersense was: Bit Resolution = 8, Sampling period = 500 ms and no average filter.

Chord	Index	Middle	Ring	Little
C	p22	p43	p54	
A		p43	p33	p23
G	p53	p64	p14	
E	p33	p53	p43	
D	p12	p13	p24	
Dm	p22	p33	p24	
Am	p51	p43	p33	
F	p61	p32	p53	p43
Bm	p51	p22	p43	p33

Figure 46: Max/MSP monitoring patch for the force readings.

The Max/MSP patch that monitors the force reading per sensor.

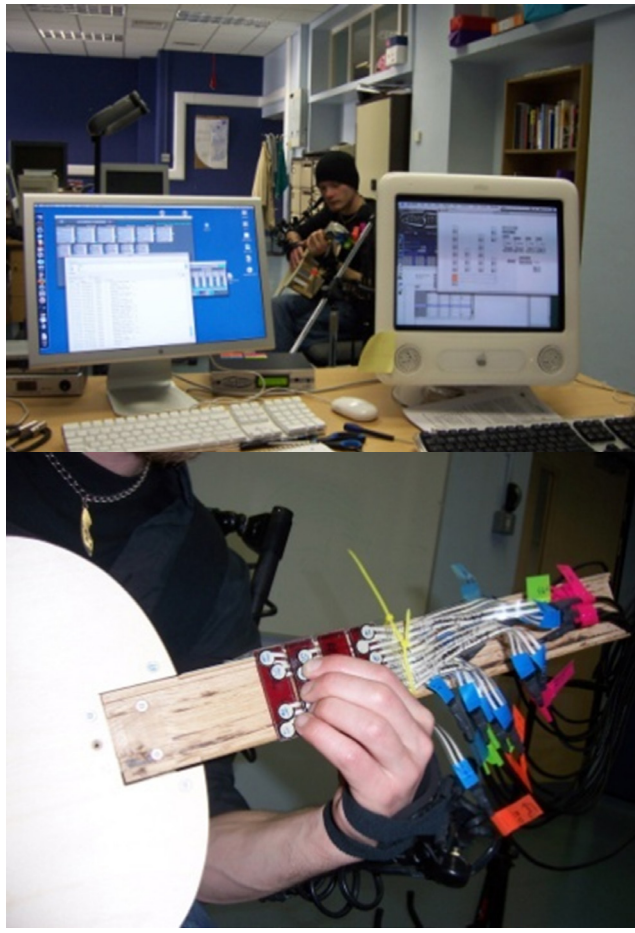


Source: Own image.

To eliminate possible fluctuations caused by external factors, all sensors were recalibrated (3 measurements) before every session

using weights of 102, 204, 306, 510, 0714, 1000 and 1.510 grams. Figure 47 shows the setup working during one of the sessions.

Figure 47: Pictures of Experiment 2 – Force and Posture.
On the figure at the top, the computer on the right side records the force measured by FoGu guitar and the computer from the left records the posture measured by the Gypsy6 skeleton. The figure from the bottom shows a close up on an A chord shape being performed in the region of the 9th fret.



Source: Own image.

Tasks

Before explaining the task a mention must be made of the use of Gypsy6 Skeleton. In this experiment, unlike the previous experiments, the subjects had to wear the skeleton, which demands individual adjustment and the calibration to the subject's body. This means that the value recorded for a particular joint, for example, wrist flexion, is relative to the flexibility of the subject for that movement (wrist flexion/extension) and a direct comparison between subjects is not possible.

The subjects, wearing the skeleton, were asked to perform the same ten chord shapes from the previous experiment in the FoGu device. They were instructed to apply the force they believed to be right to make all notes of chords sound clear. The position and force should be kept for 30 seconds. This process was repeated three times to every chord shape with a 2-minute interval between the trials.

The use of the 30-second blocks was proposed by Shan and Visentin (2003) to improve the reliability of their experiment which aimed to understand the kinematics of violin performance. Three recordings were recommended by Mathiowetz (1985) because measurements in strength studies are usually not reliable and are subject to several external inferences. The 2-minute interval is the estimated time required to recover from the 30 seconds sub-maximal muscle contraction (up to 70% MVC).

The subjects could not rely on tactile or auditory feedback as they would normally do when performing on a normal guitar. The idea is that they apply the force they are used to and not the maximal force they are capable of. Any feedback could lead them to adjust the pressure, applying more or less force than normal.

Each block of 10 chord shapes takes around 1:15 hour to be recorded. After each block, a 15 minutes break was given to the performer while the plates were repositioned to the 5th fret. The same procedure was repeated after the recording of the second block with

the consequent repositioning of the plates to the 9th fret. The total length of the experiment was 4:30 hours and the subjects were paid £30 for their participation.

Data Analysis

Even though the posture and force measurements happened at the same time, the skeleton and FoGu had no synchronisation mechanism. An algorithm was designed to synchronise the datasets but, initially, they were processed separately.

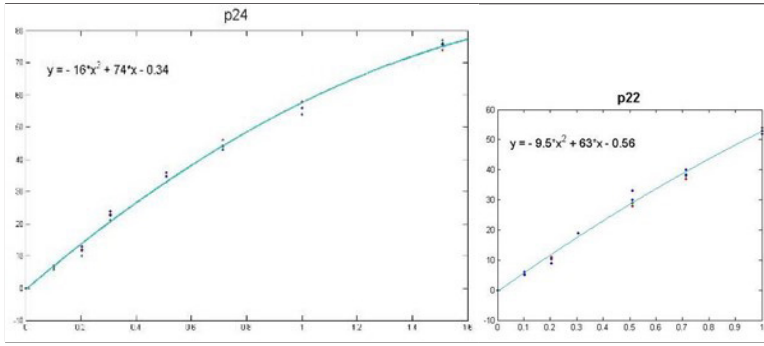
FoGu Data

At a rate of two readings per second, the total number of events for the three blocks of 30 seconds each will be 180 entries per chord x per fret x per subject.

Tekscan, the manufacture of the force sensors, claims a linearity error of 5%. However, even after conditioning the sensors (a phase required to make the output constant), this linearity could not be reached. Unfortunately, Tekscan did not offer support because we used a sensor reader (IRCAM's Ethersense interface) from a different manufacturer. Tekscan's sensor reader is limited to 16 sensors; hence we opted for the Ethersense interface.

To overcome this issue, the sensors had to be previously calibrated at least three times using weights of 102, 204, 306, 510, 0714, 1000 and 1510 grams before each session. Using a Curve Fitting Toolbox of Matlab 7.0 the model was fitted to the data and an equation for every sensor was derived. Figure 48 show some of the equations found.

Figure 48: Example of the equations derived from sensor calibration data. Every sensor has its own equation to transform its output into a kilogram force value.



Source: Own image.

The decision of which model to use was based on common sense and observation of the researcher, who is familiar with the data and aware of possible discrepancies and outliers. Norms of residuals and percentages of variance were also used to support the researcher's decision when the model seemed to provide similar fits.

A Matlab script was written to convert all the data, based on the model generated for every sensor, to the engineering unit of Kgf. As previously stated, every chord shape used a different group of sensor and every sensor served more than one finger based on the chord shape (Table 8). To facilitate the process we requested that the subject use the same fingering to perform the chord shapes.

Once all the data was processed, then Matlab and Excel were used to make the statistical analysis and generate the graphs.

Gypsy6 Skeleton

Force and posture were recorded at the same time by two different computers but without any synchronisation mechanism. The subjects were asked to perform the chord, hold their position for 30

seconds, and then repeat twice. The pauses were used as cues for rough synchronisation. The first step in data analysis was to find the cues in the MIDI data to differentiate the 'holding the chord' state from the relaxed state.

The Gypsy6 Skeleton sensors are very sensitive. They send a MIDI message every time they sense a variation of 0.125 degrees, so even the slightest movement would trigger a message. This is useful in the performance arena but it adds to the communication and complexity of the task of extracting postures from the data.

As discussed, the limbs can be configured in several ways to perform a chord shape, and the initial posture selected by the performer may adapt and change according to his sensorial feedbacks and fatigue within the 30 seconds of the trial duration. The adjustment of the angle of one articulation will force the adjustment of all the other articulations of the limb; therefore average values can not be used. Instead, all possible postures must be found so the performer's preference can be determined by the frequency these postures are performed. We call these postures Frozen Positions (FPs).

FPs can be found when all the sensors stop sending messages for a given amount of time, called Time Frame (TF). The smaller the TF, the higher the number of FPs found. To help us find the FPs in the data, we have developed a piece of software that searches the FPs with TFs of 2, 5 and 10 seconds.

We believe that a 10 seconds pause (a third of the total time) without any motion shows that the performer is comfortable in that position. If no FP of 10 seconds is found then 5 seconds FP are examined. If more then one candidate is found, then the frequency of FP of 2 seconds will be used as the criteria to choose the winner. Note that all FP are viable configurations; the winning posture is treated as the preferred but not the only posture.

Results

The statistical analysis of the data reveals many interesting hidden patterns and novelties in the way the tasks are performed but this is not our goal. The main contribution of this research is related to the computational side of it, to clarify, an algorithmic solution capable of identifying the patterns in the data and artificially simulating human constraints when playing the guitar.

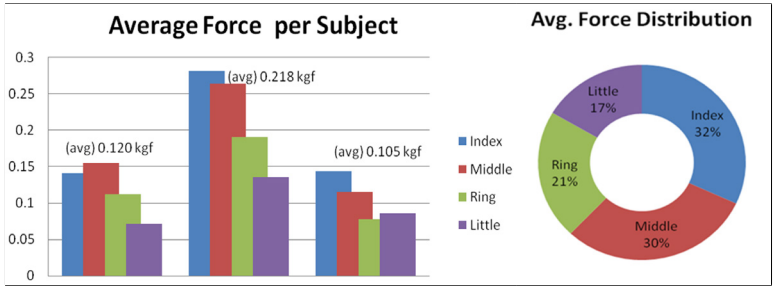
Whatever fact is revealed through the statistical analysis of the data will not be explicitly coded. These facts and patterns must be identified through machine learning algorithms, discussed in 0. The findings presented are merely a guide to verify the efficiency of such techniques.

Force

Kong and Freivalds (2003) reported that individual fingers do not contribute equally to force production. In their experiment, they found that the middle finger is the strongest at 28.7% of the grip force, followed by the index, ring, and little fingers, with percentage contributions of 26.5, 24.6, and 20.2% respectively. They believe that the middle finger has an advantage when gripping due to its central location in the hand.

The results of our force experiments have shown that, for the particular task of performing chords, the index finger is actually the strongest, contributing on average 32% of the force generated in a combined pinch grip. Note, however, that this value has been pushed up by the barre-chords, using a different type of grip (palmar pinch grip) in which the index finger is highly stressed. The middle, ring and little finger contributed 30, 21 and 17% respectively, as seen in Figure 49.

Figure 49: Finger average force distribution performing the chords.
The image on the left shows the average force of the fingers (y-axis) in kilogram force per subject (x-axis). The image on the right shows the percentage per finger of average force produced.

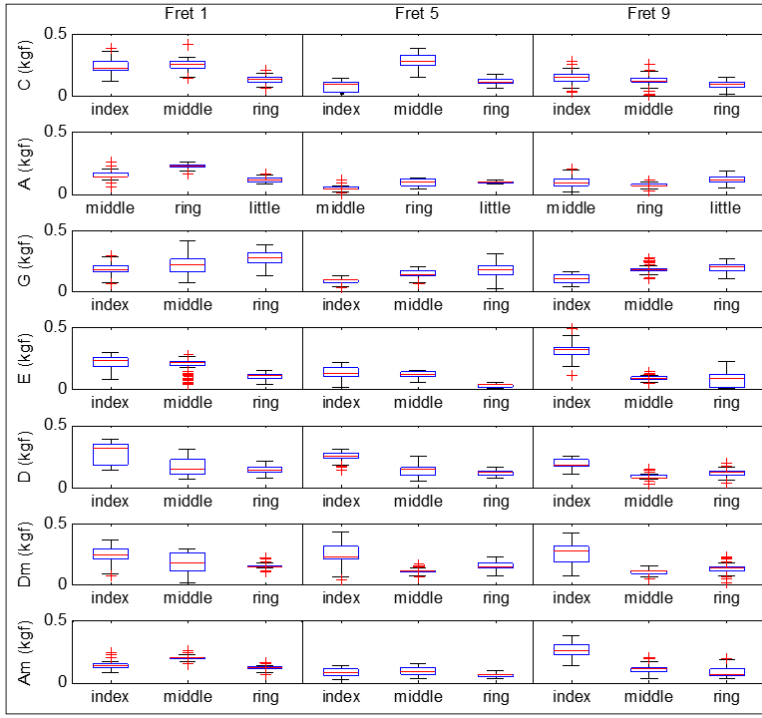


Source: Own image.

The average force the guitarists believed was necessary to produce a clear note is around 147 grams/f. Subject 2 has once again distinguished himself from the other subjects by applying double the force, on average 218 grams/f while subjects 1 and 3 have applied 120 and 105 grams/f respectively.

Since the barre-chords (B, Bm and F) and non-barre chords (C A G E D Dm Am) require two distinctive hand grips for performance, their results are plotted separately. Figure 50, Figure 51, Figure 52 show the force applied per digit to perform the non-barre chords respectively by Subject 1, 2 and 3. Note that the A chord is the only one that does not use the index finger and does use the little finger.

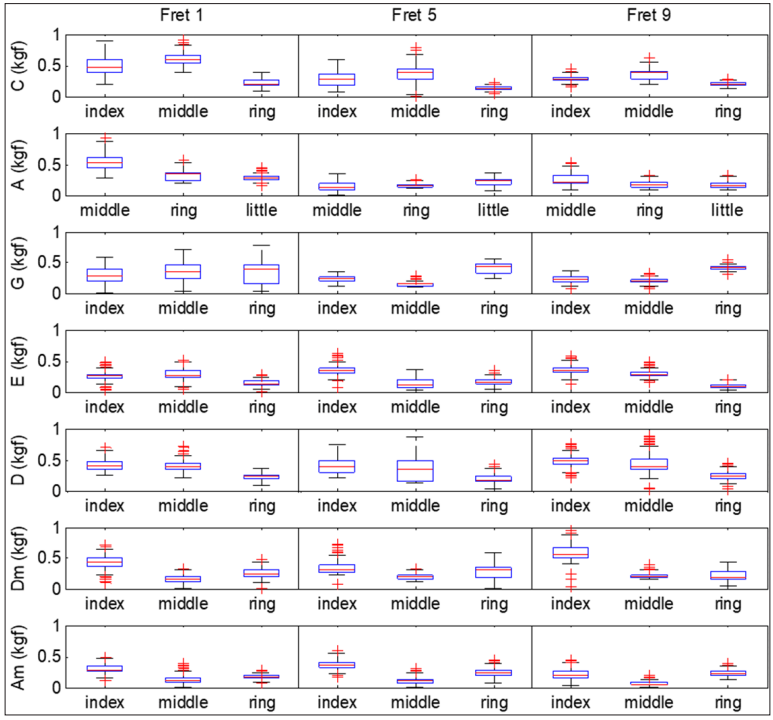
Figure 50: Non-barre chords readings for Subject 1.
The x-coordinates represent the fingers involved in the chord execution and the y-coordinates the force measured in kilogram force.



Source: Own image.

The force data is presented using the Matlab Boxplot. On each box, the central mark represents the median, the edges of the box are the 25th and 75th percentiles and outliers are plotted individually.

Figure 51: Non-barre chords readings for Subject 2.
The x-coordinates represent the fingers involved in the chord execution and the y-coordinates the force measured in kilogram force.

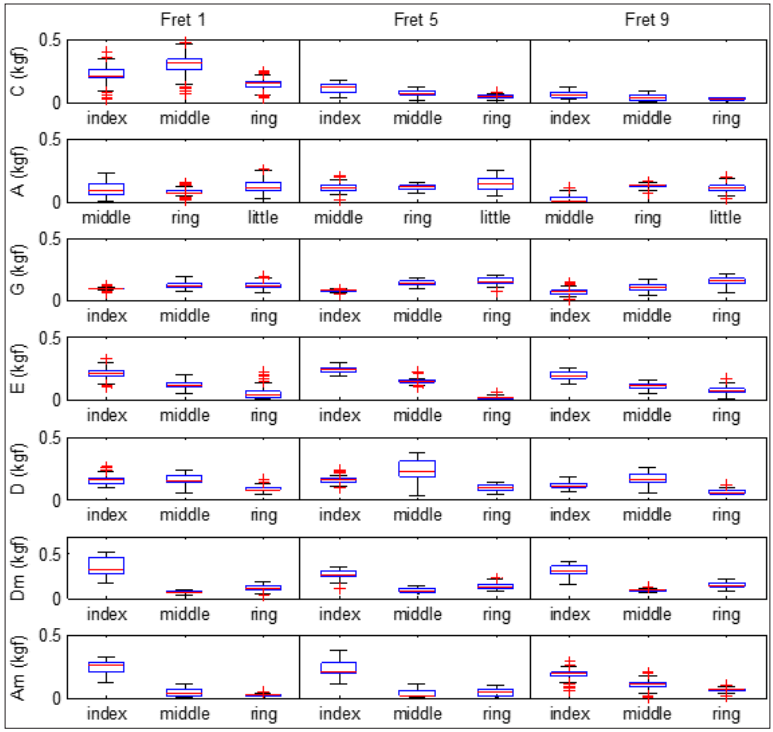


Source: Own image.

The visual analysis of the graphs can reveal patterns in the force distribution when performing the chords. For example, take chord G. All three subjects exert more force with the ring finger. It is also possible to notice that Subject 2 is more consistent in the distribution of the force among the fingers.

The supposed correlation of the fret location and force production could not be verified for the non-barre chords. The same is not true for barre-chords.

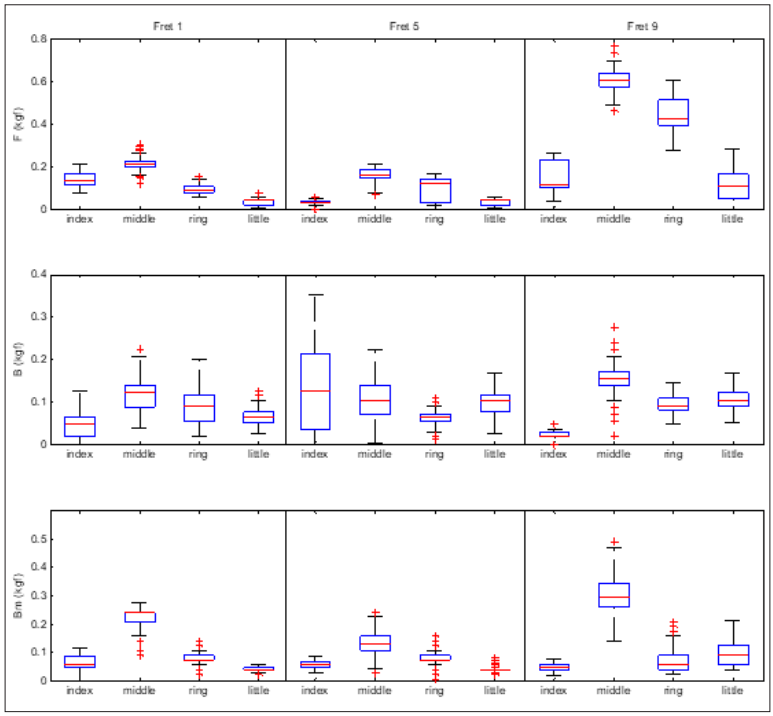
Figure 52: Non-barre chords readings for Subject 3.
The x-coordinates represent the fingers involved in the chord execution and the y-coordinates the force measured in kilogram force.



Source: Own image.

For the barre-chords, Subjects 1 and 3 presented significantly higher force production in the higher frets. Whatever the reason leading to this behaviour, it did not seem to affect Subject 2. Although merely speculative at this point, we believe that the extra-force used by Subject 1 and 3 was an attempt to overcome any difficulty originating from an awkward posture. The graphs for barre-chords are presented in Figure 53, Figure 54, and Figure 55.

Figure 53: Barre-chords readings for Subject 1.
The x-coordinates represent the fingers involved in the chord execution and the y-coordinates the force measured in kilogram force.



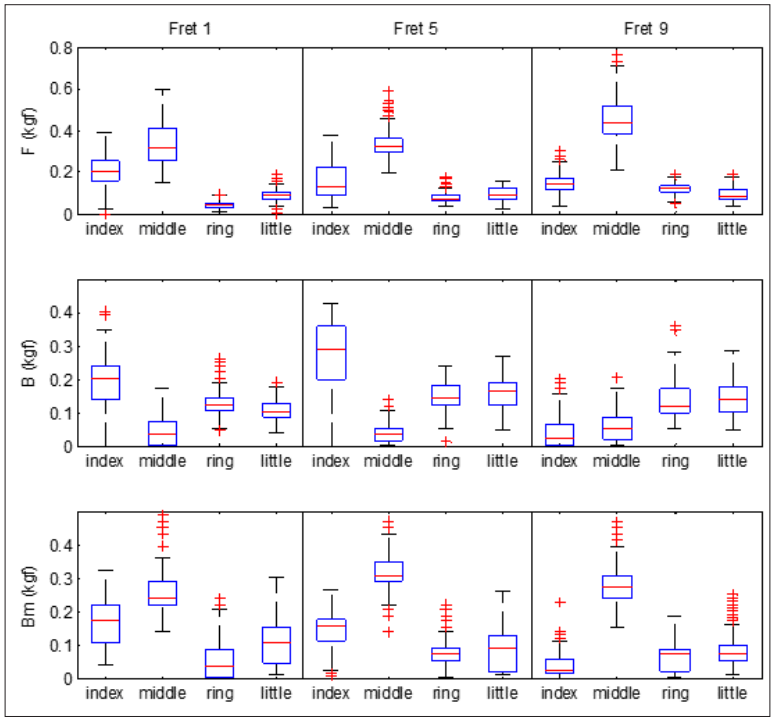
Source: Own image.

Another interesting observation is that the force produced per finger on the barre-chords is lower than on the non-barre chords. In summary, barre-chords are not only slower and less-precise but also it is more likely to produce muffled and buzzed notes.

In Figure 56 and Figure 57 it is possible to observe the force applied on every 'string' using the barre technique. Observe once again the difference in the technique of Subject 2 when compared to the others; Subject 2 manages to apply less force on the lower strings

to focus on the bass note. Meanwhile, Subject 1 and 3 apply more force on the lower strings.

Figure 54: Barre-chords readings for Subject 2.
The x-coordinates represent the fingers involved in the chord execution and the y-coordinates the force measured in kilogram force.



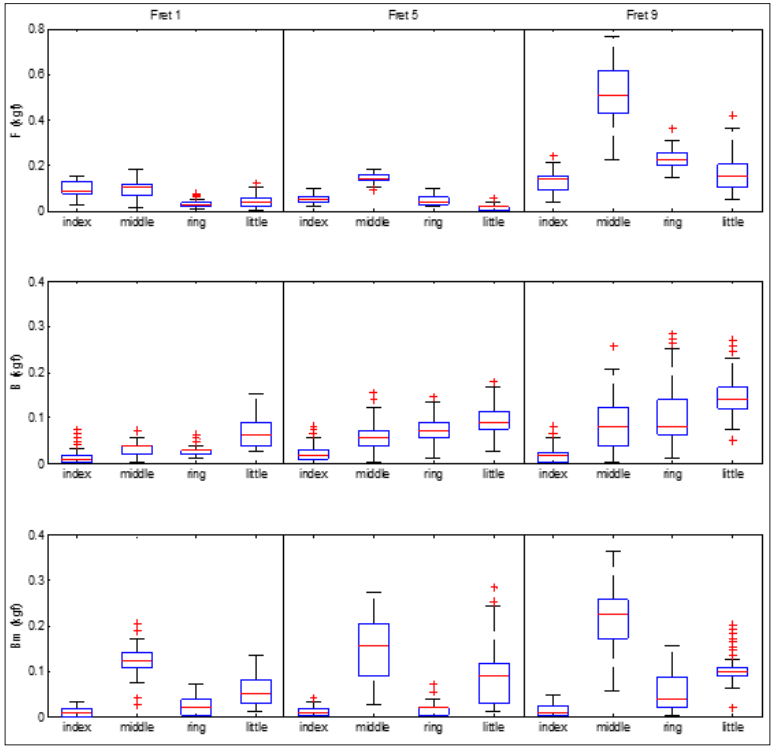
Source: Own image.

The barre technique is a way to stop several strings using only one finger. On the Bm and B chords, it is equally important to stop the 1st string and 5th string using the index finger. On the F chord, the barre should stop the 1st, 2nd, and 6th.

To produce clear notes, all strings should be stopped, however, the emphasis given to this by the performer may reveal a particular

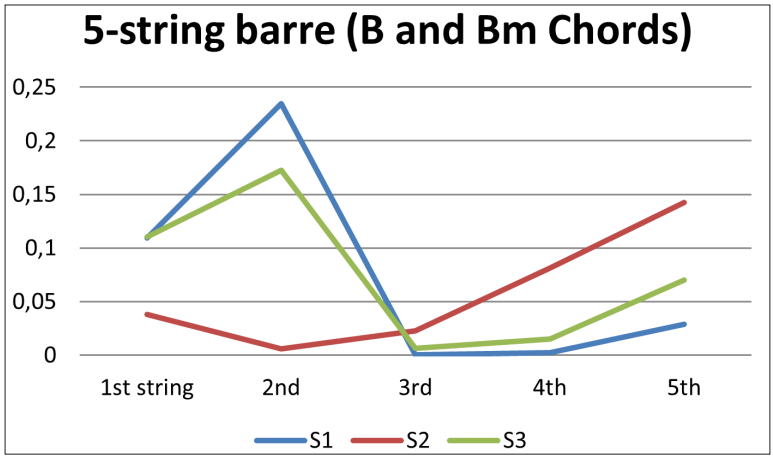
musical background and use of the right-hand technique. In addition, considering that the 6th and 5th strings (bass) are under more tension than 1st and 2nd strings, the overall technique of Subject 2 may be more efficient.

Figure 55: Barre-chord readings for Subject 3.
The x-coordinates represent the fingers involved in the chord execution and the y-coordinates the force measured in kilogram force.



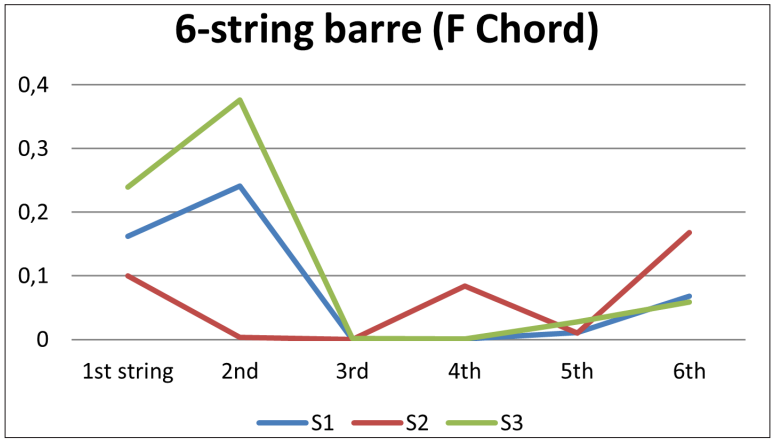
Source: Own image.

Figure 56: Barre's positions reading for B and Bm chords.
The x-coordinates show the strings pressed in the barre technique the y-coordinates the force measured in kilogram force. S1 = Subject 1, S2 = Subject 2, S3 = Subject 3.



Source: Own image.

Figure 57: Barre's positions reading for F chord.
The x-coordinates show the strings pressed in the barre technique the y-coordinates the force measured in kilogram force. S1 = Subject 1, S2 = Subject 2, S3 = Subject 3.

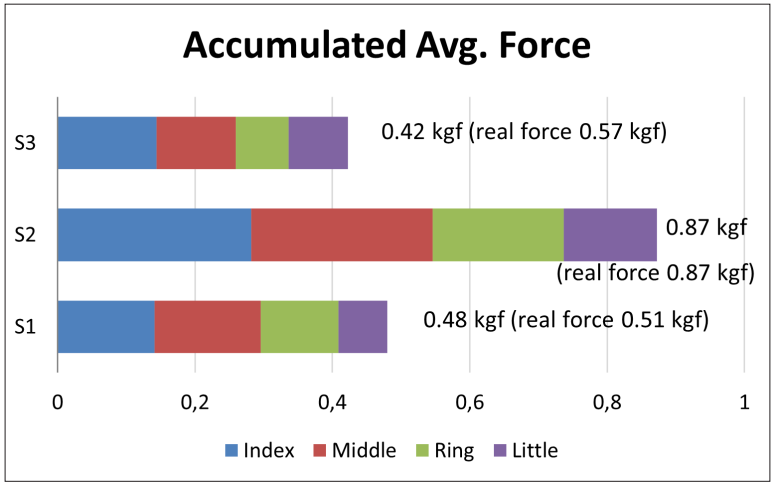


Source: Own image.

In all the previous graphs for barre-chords, the index finger has always been linked with bass-note for the calculation of the average force (i.e. in the F chord the index would plot the values of position [6,1]). However, as can be seen in Figure 56 and Figure 57, the position of the bass note is not necessarily where more force is applied.

If the computation considered the maximum force generated by the index finger regardless of its position in the barre, then the average accumulated force production for Subjects 1 and 3 would be considerably greater, as seen in Figure 58.

Figure 58: Accumulated average force.
The image shows the force participation of the finger in chords per Subject, where S1 = Subject 1, S2 = Subject 2, S3 = Subject 3. The 'real force' considers the index finger maximum force in the barre as to calculate the average, whereas the normal force considers the uppermost position in the barre.



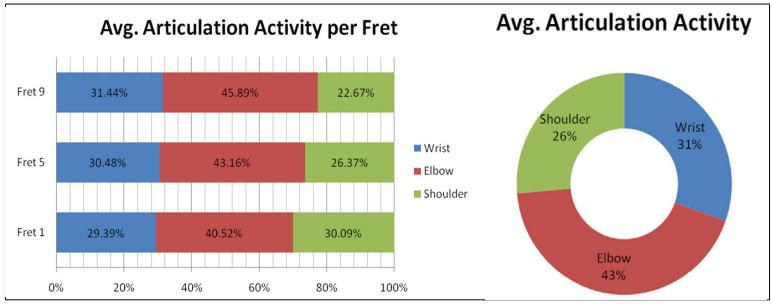
Source: Own image.

The skeleton used has allowed us to record the slightest variation in angles of the wrist, elbow and shoulder joints. The average number of messages sent by each articulation may indicate the workload needed to perform the task.

The perfect execution of a movement is the result of the balanced effort of all the joints of the limb's biomechanical system. One joint can overcome the deficiency of another up to a certain degree; however, they are all linked. There is no isolated movement when performing an instrument and finding the right balance will contribute not only towards the mastering of the instrument but will also reduce the risk of injuries related to the task.

In our experiment, the elbow was the joint that presented the highest level of motion with 43% of the overall messages, followed by the wrist and shoulder with 31 and 26% respectively (Figure 59). These results however can be misleading if we consider the number of *degrees of freedom (df)* of each articulation and the number of sensors the skeleton uses to record them.

Figure 59: Articulation activity.
The graph on the left shows the proportion of the recorded articulation activity (x-axis) per fret region (y-axis). The graph shows the overall average percentage of the recorded articulation activity



Source: Own image.

The skeleton claims to measure the following movements of the upper-limb: wrist up/down, wrist spin, elbow up/down, elbow left/right, arm up/down, and arm left-right. However, the terminology used by the manufacturer can lead to an incorrect perception of the articulation in use. For instance, the elbow left/right movement is achieved by the rotation of the shoulder in lateral/medial rotation.

Table 9: The terminology used to describe movements and articulations.

Movement	Articulation	Orientation	ROM	MIDI
Wrist up/down	Wrist	Flexion	0-90	0
		Extension	0-70	127
Wrist Spin	Elbow(forearm)	Supination	0-90	0
		Pronation	0-90	127
Elbow Up/Down	Elbow	Flexion	0-160	0
		Extension	0-145	127
Elbow Side	Shoulder	Lateral	0-90	0
		Medial	0-90	127
Arm Up/Down	Shoulder	Extension	0-50	0
		Flexion	0-90	127
Arm Side	Shoulder	Adduction	90-0	0
		Abduction	0-90	127

Table 9 shows the articulation used in accordance with the terminology adopted by the manufacturer. The ROM column shows the typical Range of Motion of the articulation by the orientation (Freivalds, 2004, p.385). The MIDI column shows the value sent by the skeleton to the appropriate MIDI channel when the articulation is stationary in either extremity of the movement.

The high level of the combined use of wrist and elbow movement might not come as a surprise to a professionally trained guitarist. Interestingly, the level of shoulder articulation decreases toward the higher frets in comparison with the wrist and elbow (Figure 59). Note that an important wrist movement (radial-ulnar rotation) is not

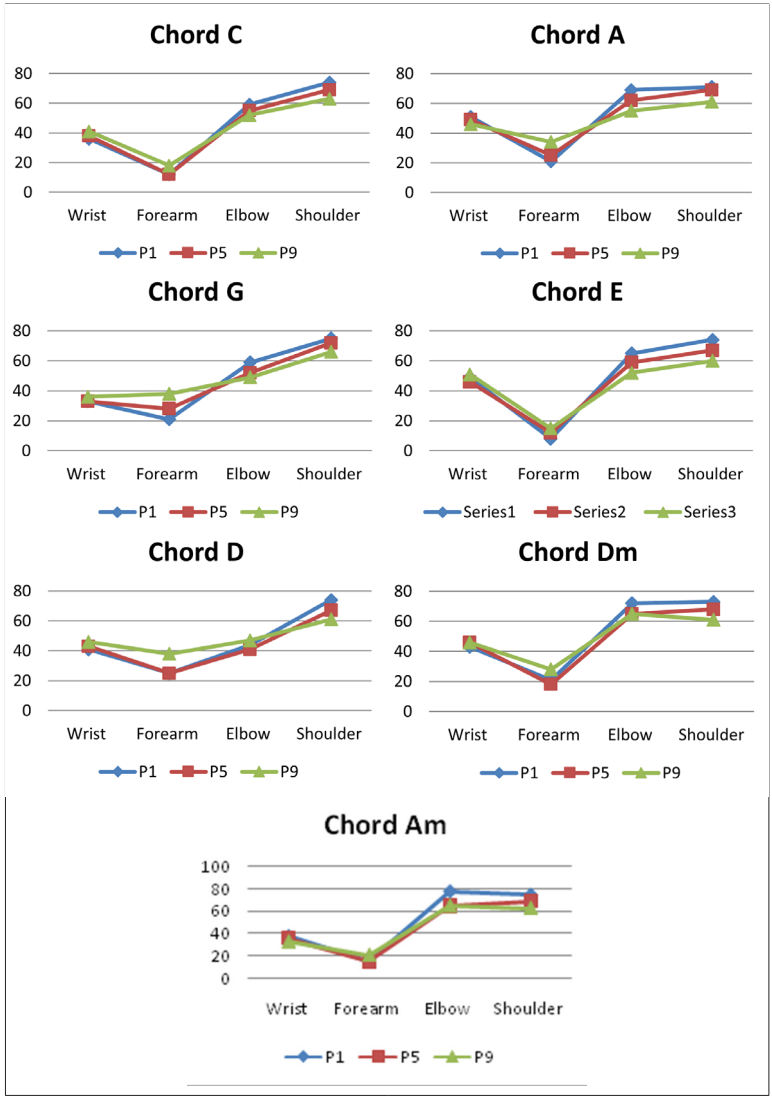
captured by the skeleton; if this movement had been recorded, the frequency of the wrist movement would probably surpass the elbow.

As previously stated, the posture data can not be used comparatively among the subjects as the skeleton is calibrated to the subject's body; there is no static reference point to measure the absolute angle of the joint. Hence, instead of comparing the absolute values of every FP (frozen position) among all subjects, we will compare the FPs of the chords of a single subject. We elected Subject 2 due to the better quality of the data recorded.

The data from the arm movements were not considered due to the low number of messages and constant noise caused by the operation of the skeleton near the limits of the joint range. Hence the 'shoulder' label seen in Figure 60 is related to the 'Elbow Side' movement. The 'forearm' label identifies the 'wrist spin' movement, also known as forearm rotation movement.

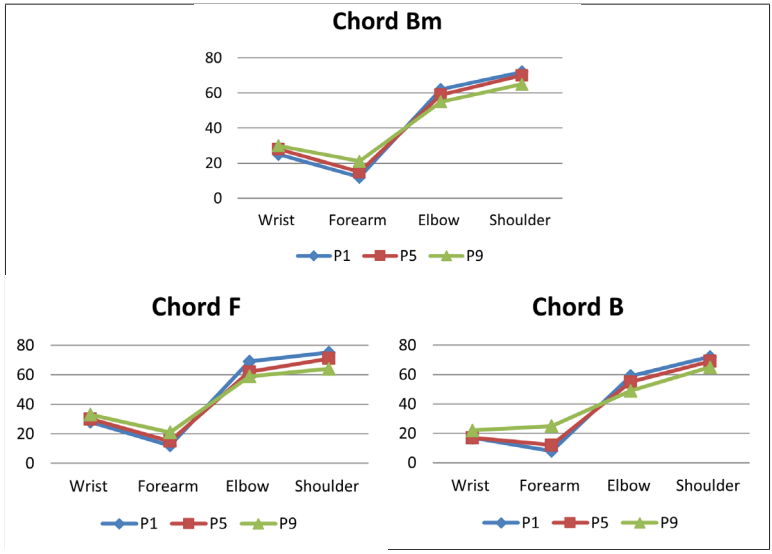
The graphs that are seen in Figure 60 and Figure 61 help the identification of similarities in the configuration of the articulations in all three regions of the fretboard. Two attributes need to be observed: a) the individual value of every movement; b) the overall configuration of the articulations given by the shape of the line.

Figure 60: Left upper-limb articulation data of non-barre chords.
The x-coordinates show the articulation and the y-coordinates the respective angle in which the articulation was configured to perform the chord (see Table 9), P1 = Fret 1, P5 = Fret 5, and P9 = Fret 9 region.



Source: Own image.

Figure 61: Left upper-limb articulation data of barre chords
The x-coordinates show the articulation and the y-coordinates the respective angle in which the articulation was configured to perform the chord (see Table 9), P1 = Fret 1, P5 = Fret 5, and P9 = Fret 9 region.



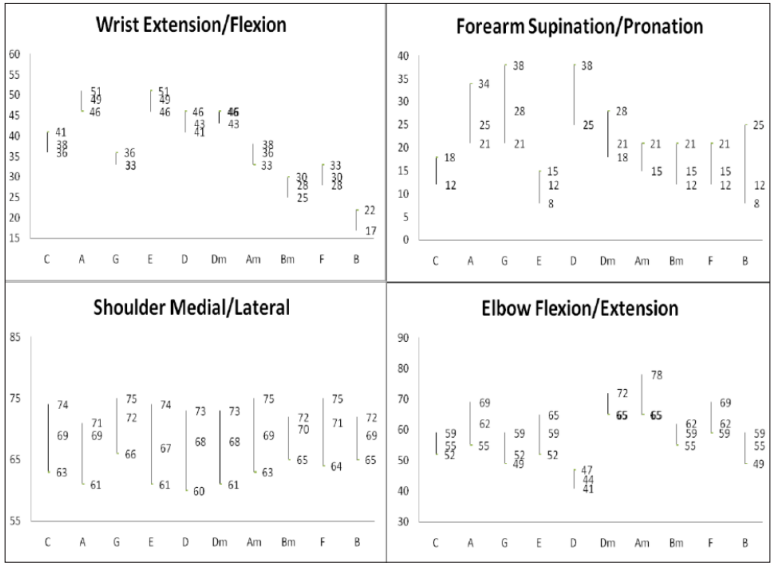
Source: Own image.

By simply analysing the shape of the lines it is possible to spot the unique form of the G chord or how much the E and C chords are 'biomechanically' similar. Of course, the values found for these articulations represent just a small part of a much larger and more complex biomechanical system. One should recall that the so-called biomechanical models for guitar fingering only considered the location of fingertips on the fretboard which, presumably translates in a handicapped travel-cost function.

Another way of comparing the biomechanical characteristics of the postures used to perform the chords is to analyse the 'intensity' of the joint deviation. For instance, in Figure 62 it is possible to see that the B chord demanded a much greater flexion of the wrist than

the A or E chords. Of course, the analysis of a single *df* of an articulation as a measure of similarity between chords can be very dangerous. The same A and E chords that presented similar wrist flexion/extension values are very different under the analysis of the elbow (forearm) supination and pronation.

Figure 62: Range of values recorded for the movements per articulation. The x-coordinates show the chords, the y-coordinates a value related to the angle of the articulation (see Table 9).



Source: Own image.

The posture data, together with the force, speed and precision data which will be used to attempt to simulate a 'human' guitar performance will be discussed in the next chapter.

SUMMARY

In this chapter, we have described our methodology to capture and analyse human performance data in the context of guitar playing.

In the ergonomic and biomechanical field, performance measurements are generally associated with measures of speed or time, measures of accuracy or error, measures of strength or force. We have measured these variables in two distinct experiments: a) speed and precision; and b) force and posture.

Three guitarists of different backgrounds and skills took part in the experiments. The tasks for both experiments involved performing simple pre-determined chord shapes that have allowed us to investigate the coordinated effort of the fingers of the left-hand.

In the first experiment, we used a guitar-like MIDI controller to measure the speed and precision of the subjects performing the chord shapes. The results for speed showed significant variance in the performing styles of the subjects.

The average time for the subjects to perform a chord was around 350 ms. The fastest chord took 248 ms with slowest taking as long as 559 ms. In general, we have found that chord shapes that made use of the little finger or that demanded a hand motion were slower. The little finger was the slowest, taking an average to 360 ms to reach its target.

Trends of errors were also analysed; the fastest subject produced the most errors. Only barre chords presented errors. 51% of total errors were generated during the performance of the B chord, followed by Bm and F chords, with 41 and 8% respectively. 41% of total recorded errors were related to hitting above the string targeted.

For the second experiment, we designed and built our own measuring apparatus to record the force produced by the fingers. To validate the force measurements, the configuration of left upper-limb joints was also recorded using a Gypsy6 Exo-skeleton.

The results show that the index finger is the main force producer, contributing 32% of the overall force generated. The middle, ring and little finger contributed 30, 21 and 17% respectively. The average force produced by each finger was 147 grams/f.

To an extent, the results obtained support the hypothesis that biomechanical constraints can indeed impact the way the guitarist performs leading to performance errors. In due course, this will assist in the development of a computer-generated guitar performance with a human-feel on it. In the next Chapter, we will discuss the computational techniques we have used to achieve that.

Chapter 5

Guitar Performance Modelling

> Error is discipline through which we advance. (William Ellery Channing).

The idea of developing computational models of music performance dates back to the first computer applications. These first models were mainly dedicated to music production and experimentation (Gareth and Curtis, 1985).

Like any other musical application, these models had to handle common musical elements, such as a note, duration ratios between successive notes, ascending lines, melodic leaps, melodic and harmonic change, phrase structure, et cetera. Although these elements are essential to the formalisation of any musical application, each developer used to implement their own solution based on the particular needs of the application. It did not take long for them to realise that it would be more productive to have generic library of musical structures which they could reuse to write their application.

All the main programming languages started to provide libraries with some support for music production, even if it is as basic as a note production based on a frequency. This, however, was not enough and in the mid 1980's dedicated programming language for music started to be produced. One of the first languages to become popular was the HMSL – Hierarchical Music Structure Language (Polansky and Rosenboom, 1985; Polansky *et al.*, 1987).

A computer programming language presents an abstract model of computation that allows one to write a program without worrying about details that are not relevant to the problem domain of the program (McCartney, 2002). These languages are designed to provide a set of abstractions that makes expressing compositional ideas as easy and direct as possible (McCartney, 2002). Examples of such languages are CSound (Vercoe, 1986), MAX/MSP (Puckette, 2002), SuperCollider (McCartney, 2002), Nyquist (Dannenberg, 1997) to name a few. Each specialised into some type of musical task: composition, performance, synthesis and so on.

Still, there is one particular area that has not been explored by these languages: musical performance modelling. When a language or application claims to be designed for musical performance purposes, it is purely a performance tool that is controlled in performance--time by the performer, similar to a musician playing an instrument. This means that the performance actions are still the performer's responsibility and not something embedded in language, as one would expect in a performance 'modelling' language. In Section 2.5 (p. 42) we have seen an example of that with the PwSynth and ENP (Laurson, 2000; Laurson *et al.*, 2001).

In a more conventional setup, composers transcribe musical ideas into a written score in a way it can be understood and interpreted by another human, the instrumentalist/interpreter. Ultimately, it is the interpreter's role to convey all the emotion intended by the composer to the listeners. This is a task that requires a great deal of sensibility and intelligence that is difficult to be mimicked by a machine.

To perform a similar task, a machine would require a much more detailed specification of the music performance, however as Gareth and Curtis (1985, p.237) explains, there is just a certain amount of information that can be formalised.

> The importance of formal thinking, in music or in any other area, cannot meaningfully be separated from the development of formal languages through which that thinking is officially expressed, and unofficially explained.

In music, the active explicit creation of formal languages which is used to express aspects of theory, performance, or composition, did not come until the computer made it possible to automate some aspects of the processing of formal languages (Gareth and Curtis, 1985). The limitations of rules as constraints to a purely formal approach leave some gaps in the domain of creative, artistic (and ipso facto informal) endeavour.

This is the point where Artificial Intelligence (AI) meets Music Performance. Computers can now perform musical tasks that were formerly associated exclusively with naturally intelligent musicians (Roads, 1985). However, most of the Expressive Music Performance models (EMP) do not consider the bodily limitations behind performance actions. Even if they did, it would not be possible to formalise these physical actions using a conventional programming language for music performance because they do not provide support for this type of modelling.

To exemplify what has already been discussed, suppose we want to model a G note using a guitar. In a normal language for music performance, we have to select one of many available guitar timbres, the note's frequency (G4), intensity (velocity), and duration.

In a language for music performance modelling, we would probably have to specify among other things: the string and fret used, the string tuning, gauge and tension, the guitar scale length, the fretboard

inter-fret spacing, the finger used to stop the string and the force applied to do so, the finger used to pluck the string, nail the plectrum shape and hardness, the direction of the stroke, its intensity and region.

The example above shows that such approach is just not practical from the programmer's point of view. If this level of detail is going to be modelled, then the language needs to provide a way to do it effortlessly. This, however, is just part of the problem. One must also find a Sound Generation Unit capable of taking full advantage of all of this detailed information and produce a realistic performance.

In summary, in order to model the results of the performer's physical actions during a musical performance, two main fronts of development need to be approached:

1. A programming language/library that offers the level of abstraction necessary to detail the performer and the instrument constraints as well as supporting common musical tasks;
2. An intelligent algorithm capable of learning and predicting the biomechanical limitations of the human body during a music performance. This algorithm together with the environment mentioned above would save the programmer the time-consuming task of having to specify every single task of the performance;

Each of these two topics has its own challenges and they are described in the first two sections of this chapter. The third section will discuss the integration of the machine learning algorithms with the Octopus Music API: a Java library designed to model musical performances.

OCTOPUS MUSIC API (APPLICATION PROGRAMMING INTERFACE)

Computers have been used to perform musical-related tasks in many different areas such as audio signal processing, score representation, compositional assistance, and real-time control of the complex

processes that go into creating, performing, and analysing music. However the development of programming languages specifically for musical applications seems to have concentrated on the areas of sound synthesis and musical composition (Loy and Abbott, 1985).

According to Loy and Abbott (1985), three strategies have been commonly employed in the development of musical tools:

1. modifying a composition program written in an existing programming language;
2. writing a programming language as the embodiment of a musical paradigm;
3. developing libraries of utility subroutines that implement common operations on musical data structures then writing composition programs in some standard programming language

The Octopus Music API is an example of the third approach and, as one can imagine, it is not the only programming library with a musical purpose. Numerous software packages have been written for applications in music composition, music analysis, sound synthesis, and sound manipulation (Pennycook, 1985). Unlike other Java APIs' for musical software development, the Octopus Music API is specifically designed to model the interaction of the musical performance elements, mainly the performer and musical instrument.

As a programming library, the Octopus can be used to write any application that requires dealing with musical structures, such as composition or musical educational software. As an example of the possible use for the Octopus Music API, suppose that software has to play a simple harmonic progression composed of the chords 'C – F7 – G'. This is a very simple task; all that the software has to do is to determine the notes of the chords and play them in a specified tempo and timbre (i.e. GM guitar). Any musical programming language can do this with ease. This is what we call the 1st layer of abstraction.

Now, let us add another feature to this software. The harmonic sequence needs to be played with the 'Admira Concerto Classical' guitar timbre (standard tuning) using a particular chord fingering.

In addition, a particular guitar strumming should be applied. This 2nd layer of abstraction will require a base (harmonic) guitar playing knowledge and the ability to communicate with a Sound Generation Unit able to render the particular timbre of that guitar. This layer is much more specialised than the first, but still possible to be achieved using most musical programming languages as long as the programmer is experienced and is prepared to work hard.

The 3rd layer is where the specialisation reaches another level. The software is now requested to play the same sequence using not only the timbre of 'Admira Concerto Classical' but also considering its playability and mechanics (string gauge, dimensions, tuning, etc). Additionally, the sequence should be played by a certain flamenco guitarist named B.B Queen who is left-handed and famous for his peculiar use of the rhythmic-hand; B.B Queen likes to play the guitar placed on his right leg and he is slightly anxious with this gig because he never played this particular guitar before. Facing a demanding audience, B.B Queen is wondering if he should have rehearsed more rather than dedicated his last 3 months trying to learn wakeboard in the Caribbean Islands.

The use of layers illustrates the different levels of complexity, and consequently the amount of coding required to achieve the goals set. Imagine the amount of recoding that would be necessary if the guitar or performer had to be replaced. Even worse, imagine if the harmonic sequence has to be played on a different instrument or even in an 'extended instrument' [4]without a formal performance technique.

The Octopus Music API was designed to deal with the 3rd layer making the most of the Object-Oriented Programming (OOP) concepts, such as encapsulation, polymorphism, and inheritance (Booch *et al.*, 2007). The API was organised in a way that contemplates its extension so in the future, it could be extended to model all sorts of musical performance. The Octopus API design is explained in the next section.

---

4 Extended instruments are acoustic instruments equipped with sensors and other customised electronic components designed to extend the instrument capabilities.

Octopus Project Design

One of the first decisions to be made once the full requirements of a software (including a library) have been identified is the selection of the most suitable technology for the job, in this case, an Object-oriented Programming (OOP) language with basic support to audio and MIDI functionalities. We have decided to use the Java SDK (version 1.5).

One could argue that, compared to languages specially designed for Computer Music, Java suffers from a slower performance. Indeed Java sacrifices performance in order to be flexible and portable, allowing the integration of music and sound processing with features like networking, graphics rendering, mobile devices programming, database, etc. (Costalonga et al., 2005). Furthermore, Java is free and widely used in academics.

The Octopus Music API is composed of 80 classes (52 publics) totalling over 16,000 lines of code. The packages are physically organised as follows:

Package octopus: The classes in this package represent general musical structures, such as *Note, Chord, Melody*, etc.

Package octopus.instrument: This package contains general classes that are useful to all musical instruments modelled within Octopus. The classes on in this level must be generic enough to allow the expansion of the API to new instruments.

It is in this level that a more abstract musical structure such as *Melody* (package octopus) becomes the more practical *PerformableMelody*, so rather than only notes in a musical score, the *PerformableMelody* contains instructions of how to play the *Melody* in a particular *Instrument*.

Package octopus.instrument.fretted: These are specialised classes for fretted instruments, normally from the Luto family. It is at this level that we solve some problems related to guitar performance modelling. For instance, how to pinpoint a note position in the guitar fretboard (class *GuitarNotePosition).*

Package octopus.communication: These classes prepare Octopus for the next step of the research: integration with a Sound Generation Unit capable of rendering a performance with the level of detail that Octopus allows. These classes work as a middle-tier communication protocol to external devices. They are internal descriptors that can be easily parsed to whatever communication protocol the device (and Java) supports such as MIDI or OpenSound Control (OSC).

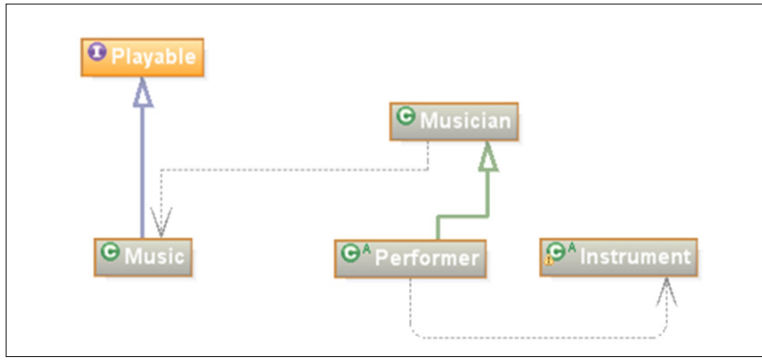
Package octopus.communication.midi: Communication classes *parse* the internal performance descriptor into MIDI messages.

Package octopus.util: Miscellaneous utility classes;

More important than the physical arrangement of the classes into packages is the classes' conceptual classification. Four categories are used:

1. Musical Data Structures classes (playable): Classes that represent musical concepts and can be played;
2. Musical Data Interpretation Classes (musicians): Classes capable of translate 'playable' objects (instance of a class) into sound;
3. Instrument Classes: Classes that allow the modelling of the musical instrument playability attributes (mechanical/ergonomic);
4. Communication classes: Bridge between the Musical Data Interpretation classes and the Sound Generator Units;

Figure 63: Simplified class diagram for the Musical Interpreters.



Source: Own image.

Figure 63 illustrates with a sample class diagram the conceptual organisation of the classes. The *Music* class (Data Structure) realises the *Playable* Interface and, as a result, becomes 'playable' to the Musician (Musical Data Interpretation). The *Performer* is a subclass of *Musician*, therefore inherits all the musical 'knowledge' which is extended to contemplate the *Instrument* handling.

The main classes of each of these categories will be discussed in the next sections.

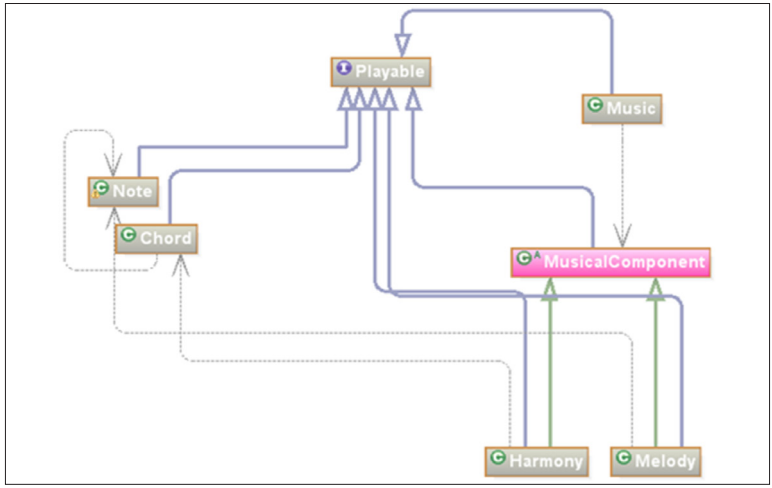
The full documentation and API (including tutorials) can be found online at http://sourceforge.net/projects/octopusmusic/.

Musical Data Structures

Generative composition languages usually come with descriptive musical data structures but emphasise compositional processing (Loy and Abbott, 1985). With many aspects of music, we know what to represent, but the issue is how to represent it (Dannenberg, 1993).

Musical Data Structures are a computational formalisation of musical concepts that are compatible with other classes of the API. It includes basic classes such as *Note, Melody and RhythmPattern*. These classes could be used as part of a hard-coded composition, as part of an application that generates music algorithmically, or any other application that benefits from the structural relations of musical elements.

Figure 64: Class diagram for Musical Data Structure.



Source: Own image.

Figure 64 presents a class diagram of some of the structures modelled in the API. All the Music Data Structures must realise the Interface *Playable*, meaning that it can be played by a *Musician*. The most obvious examples are: *Note*, *Chord,* and *Music*. A less trivial *Playable* structure is the *MusicalComponent*, which is the abstract class that any Musical Structure must implement in order to be compatible with *Music* internal data structure. Although *Note(s)* and *Chord(s)* are found in *Music*, they must first be grouped into *Harmony* and *Melody* structures that implement the *MusicalComponent*.

Class octopus.Note

Most computer music notations define a musical note as the specification of an acoustic event. In the traditional music notation, a *Note* specifies a human gesture toward an instrument (Loy and Abbott, 1985). For us, the *Note* is the smallest audible element that can be intentionally played or grouped in a musical structure.

Although the *Note* is the simplest musical element of the API, it has attributes like any other object in the OOP paradigm. The notes attributes are:

- Name: C Sharp;
- Symbol: C#
- Pitch Value: 64 (midi);
- Octave: C4.
- Accidents: (sharp, double sharp, flat, double flat);

It would be unproductive if every time a Note is required the programmer had to fill in values to all these attributes. So instead, we used a software design pattern known as Factory.

Factories are static classes that return highly demanding objects in a simple form, reducing code overhead. The factory class used to create and perform computations over Note(s) objects is the NoteFactory. Code Example 1 shows two ways of instantiating Note objects.

Code Example 1: Instantiation of a Note object using the NoteFactory static class.

```
Note A = NoteFactory.getA( );
Note Ab = NoteFactory.get("Ab");
```

Class octopus.Chord

A *Chord* is a set of *Notes* played together or *arpeggiated*. There are several ways to instantiate a *Chord* object but the recommended one is based on the chord musical notation. The chord musical notation can be seen as a language with a well defined semantic and syntax to describe *Chords*. The API, just like a compiler, runs a lexical analysis over the text describing the *Chord* and validates or refuses based on the alphabet in use. (More information in (Costalonga *et al.*, 2008)

Unfortunately, the chord notation is not standardised all over the world, meaning that the chord names (symbols) might vary among

different communities of musicians. The default *ChordNotation* using the API is based on Brazilian Bossa Nova musical genre known by its complex harmonies. If the *ChordNotation* is not adequate for software that is being developed than it might be necessary to load another notation (file). An example of *Chord* instantiation can be seen in Code Example 2.

Code Example 2: Chord instantiation.

```
Chord chord = new Chord("C#m7(add11)");
```

The chord object in Code Example 2 is populated by the *Note(s)* objects linked to the *Interval* that describes their role, as shown in Table 10.

Table 10: Chord's notes.

Index	Interval	Note
0	Fundamental(root)	Note C#
1	Minor 3rd	Note E
2	Perfect 5th	Note G#
3	Minor 7th	Note B
4	Major 11th	Note F#

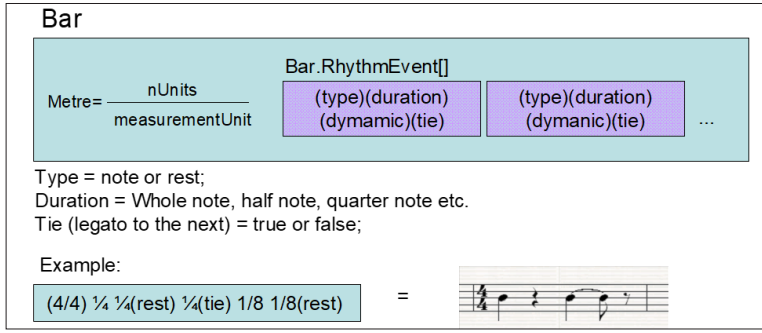
Two additional pointers are used to indicate the root and the bass note, which is not the same note in inversion cases. In the example given, both pointers are on *index 0*.

Class octopus.Bar

A *Bar* is simply a rhythmic phrase. It is a collection of the smallest rhythmic structure designed in the API. The *Bar.RhythmEvent* can be either note or rest with values between 0 and 1 for the duration, dynamic, and accentuation. The tie attribute is used to indicate whether the duration of the *RhythmEvent* should be linked with the next one in the sequence.

The interaction between real-time, measured in seconds, and metrical time, measured in beats, is frequently addressed in music representation schemes (Dannenberg, 1993). The time signature of the *Bar* is written in the form of a fraction given by the number of rhythmic units divided by the measuring of the unit. The *Bar* will not prevent the input of *RhythmEvents* that exceeds the time signature instead, the *bar.getSignatureDistance( )* method was implemented to inform the programmer if the R*hythmEvents* are in accordance with the *metre* or not. Figure 63 shows the internal structure of the Bar with an example.

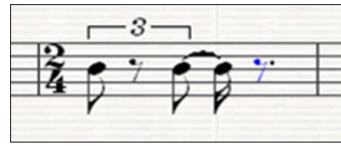
Figure 65: Bar internal data structure.



Source: Own image.

RhythmEvents do not always have to obey the metre. A *tuplet* allows the organisation of two of more *RhythmEvents* in a time frame (duration) smaller than the total duration of the events in the *tuplet*. Code Example 3 shows how to implement the *tuplet* illustrated in Figure 66.

Figure 66: Example of the score representation for a Tuplet.



Source: Own image.

Code Example 3: Tuplet.

```
Bar bar = new Bar(2,4);
   bar.addRhythmEvent(Bar.QUARTER_NOTE.Bar.RHYTHM_EVENT_NOTE);
   double[ ] reDurations = {Bar.EIGHT_NOTE ,
                            Bar.EIGHT_NOTE,
                            Bar.EIGHT_NOTE };

   int   [ ] reTypes = {Bar.RHYTHM_EVENT_NOTE ,
                        Bar.RHYTHM_EVENT_REST,
                        Bar.RHYTHM_EVENT_NOTE};

   boolean isTie = true ;
   double tupletDuration = bar.QUARTER_NOTE;

   bar.addRhythmEvent(reDurations, reTypes, tupletDuration , isTie);
   bar.addRhythmEvent(Bar.SIXTEENTH_NOTE, Bar.RHYTHM_EVENT_NOTE);
   bar.addRhythmEvent(bar.getDottedValue(Bar.EIGHT_NOTE),
                                Bar.RHYTHM_EVENT_REST););
```

Class octopus.RhythmPattern

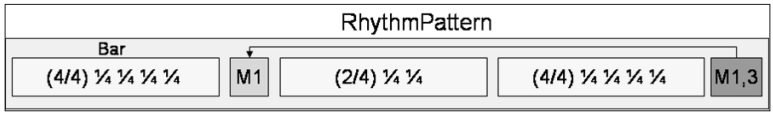
In the Octopus Music API, the rhythmic line is defined independently of the *Melody* or *Harmony* and it is represented by the *RhythmPattern.*

Both the *Melody* and the *Harmony* can be linked to *RhythmPattern.* In *Melody*, the *RhythmPattern* is mapped to the *Notes* while in *Harmony* it is mapped to the *Chords.* If the *ChordNotes* required individual rhythmic manipulation (different start time and/or duration) then the *Arpeggio* Class must be used, as explained in the next section.

The *RhythmPattern* is composed of *Bars*, *Marks* and *Returning Points.* The *Bars* are inserted sequentially so the order of input must be observed. Like the *Bar*, a *Mark* is placed in a certain position of the *RhythmPattern.* Every time that a *ReturnPoint* is reached, the pointer goes back to its respective *Mark.* This loop lasts while the number of repetitions specified in the *Return Point* is not achieved.

Figure 67 shows the internal structure of a *RhythmPattern*. In this particular example, there is a returning point placed after the third *Bar* that goes back to *Mark* M1 three times. Code Example 4 defines the *RhythmPattern* illustrated by Figure 67.

Figure 67: *RhythmPattern* internal data structure.



Source: Own image.

Code Example 5: RhythmPattern.

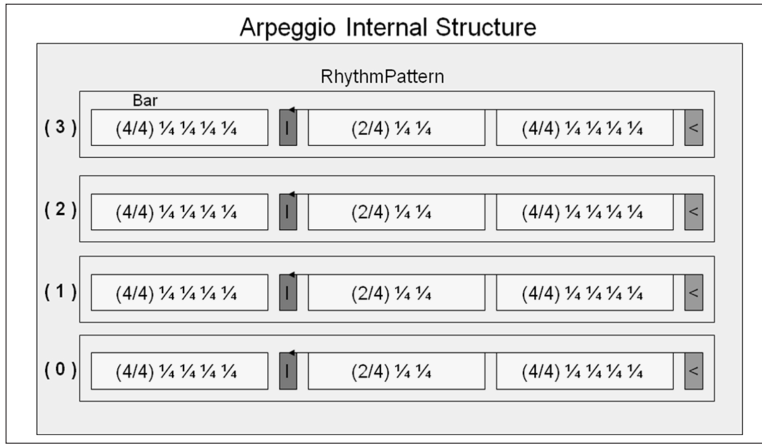
```
RhythmPattern rhythmpattern = new RhythmPattern();
 Bar bar1 = new Bar(4,4);
  bar1.addRhythmEvent(Bar.QUARTER_NOTE,1);
  bar1.addRhythmEvent(Bar.QUARTER_NOTE,1);
  bar1.addRhythmEvent(Bar.QUARTER_NOTE,1);
  bar1.addRhythmEvent(Bar.QUARTER_NOTE,1);

Bar bar2 = new Bar(2,4);
  bar2.addRhythmEvent(Bar.QUARTER_NOTE,1);
  bar2.addRhythmEvent(Bar.QUARTER_NOTE,1);

Bar bar3 = new Bar(4,4);
  bar3.addRhythmEvent(Bar.QUARTER_NOTE,1);
  bar3.addRhythmEvent(Bar.QUARTER_NOTE,1);
  bar3.addRhythmEvent(Bar.QUARTER_NOTE,1);
  bar3.addRhythmEvent(Bar.QUARTER_NOTE,1);

//placing the bar and setting the return point
  rhythmpattern.insertBar(bar1);
  rhythmpattern.insertMark("M1");
 rhythmpattern.insertBar(bar2);
 rhythmpattern.insertBar(bar3);
 rhythmpattern.insertReturn("M1",3);
```

Class octopus.Arpeggio

An *Arpeggio* is a set of *RhythmPatterns* played simultaneously; it is used to spread the notes of the *Chord* throughout its overall duration (voicing). Often the *Arpeggio* information is omitted in more popular musical notations (i.e. guitar tablature) and its use varies upon to the technique and expressivity of the *Performer*.

Inside the *Arpeggio*, the *RhythmPatterns* (called voices) are organised in vertical parallel lines, as seen in Figure 68. The lowest voice (index 0) is linked to ChordNote with the lowest pitch, the second--lowest to the second-lowest ChordNote pitch and so on.

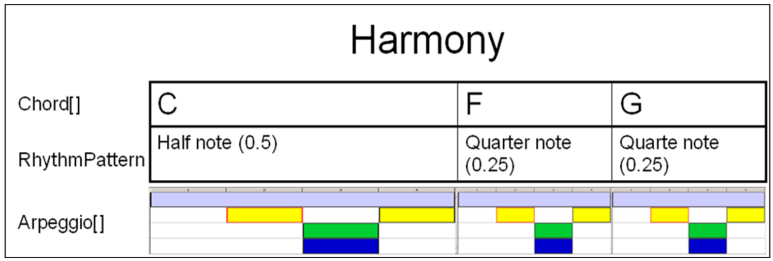
Figure 68: Class Arpeggio internal data structure.



Source: Own image.

When a *Musician* (Musical Data Interpretation Class) is requested to play a *Harmony* using a particular *Arpeggio*, it will adapt the *Arpeggio* to the *Harmony*, repeating or stretching its duration to match the duration of the *Chords*. Figure 69 illustrates the 'time stretching' feature. Note that even though the *C Chord* uses the same Arpeggio as *F* and *G chords,* its duration is twice as long.

Figure 69: Arpeggio time-stretching feature applied to the Harmony.



Source: Own image.

Code Example 6: Arpeggio.

```
Arpeggio arpeggio = new Arpeggio(4);
arpeggio.setName("Fast Arpeggio");
Bar[] bars = new Bar[4];
for(int i = 0; I < 4;  i++){
  bars[i] = new Bar(4,8);
    bars[i].addSingleRhythmEvent(Bar.EIGHT_NOTE,Bar.NOTE,i);
  arpeggio.insertBar(bars[i], i);
```

Code Example 6 shows an example of an *Arpeggio* with 4 voices, each of them composed of a 4/8 *Bar*.

Class octopus.Scale

The scale is a set of *Notes* that maintains a pre-determined interval (mode) between them. The *Scale* class allows the programmer to automatically instantiate several notes of a diatonic (Code Example 7) or pentatonic scale, which could later be used as melodic fragments.

Code Example 7: B Major diatonic scale.

```
Scale s = Scale.getDiatonicScale(NoteFactory.getNote("B"),
                                 Scale.MODE_MAJOR);
```

Class octopus.HarmonicProgression

A HarmonicProgression is to *Chords* what a *Scale* is to N*otes*. It is a set of *Chords* generated according to the degrees (given by Roman numeral) of a user-defined harmonic progression and its key.

Code Example 8: HarmonicProgression in C (key) composed by the chords respectively represented by the tonic, supertonic (minor), and dominant seventh degrees.

```
HarmonicProgression harmonicprogression = new HarmonicProgression("I -ii -V7");
harmonicprogression.addScaleDegree("I");
harmonicprogression.addScaleDegree("ii");
harmonicprogression.addScaleDegree("V", IntervalFactory.getMajorSecond());
Chord[] chords =  harmonicprogression.getChords(NoteFactory.getC());
```

The *HarmonicProgression* class allows the programmer to automatically instantiate several *Chords* following a harmonic structure in any key. Code Example 8 shows how to model a Jazz harmonic progression in the key of C.

Class octopus.Melody

Melody is a set of *Notes* played sequentially according to a certain *RhythmPattern*. The Code Example 9 creates a simple melody.

Code Example 9: Melody.

```
RhythmPattern rp = RhythmPatternLibrary.getConstantRhythmPattern();
String[] notes = {"C", "E", "F", "G", "F", "C","C", "E", "F", "G", "F", "C"};
Melody melody = new Melody(notes,rp);
```

Class octopus.Harmony

Harmony is a set of *Chords* played sequentially according to an overall *RhythmPattern* but respecting the *Arpeggios assigned to each chord.*

If an *Arpeggio* is not assigned to a *Chord* then all the *Notes* of the *Chord* will sound simultaneously and lasts for the duration of the *Chord.* The duration of each *Chord* is the same as the *RhythmEvent* associated with it, as previously illustrated in Figure 69.

Code Example 10: Coding a *Harmony* with 3 Chords that uses two different *Arpeggios.*

```
// Instantiate a harmony object with a demo RythmPattern  ;
Harmony harmony = new Harmony(RhythmPattern.getDemoRhythmPattern());

// Creates the chords of the harmony
Chord[] chords = new Chord[2];
chords[0] = Chord.getChord("C");
chords[1] = Chord.getChord("F");
Chord chord = Chord.getChord("G");

// Creates the arpeggio to the vector of chords.
Arpeggio arpeggio1 = ArpeggioLibrary.getDemoArpeggio();
arpeggio1.setTimeStratch(true);

//Creates the arperggio for G chords
Arpeggio arpeggio2 = ArpeggioLibrary.getDemoArpeggio2();

//Assing the chords to the harmony.
harmony.addChord(chord, arpeggio1);
harmony.addChord(chords, arpeggio2);
```

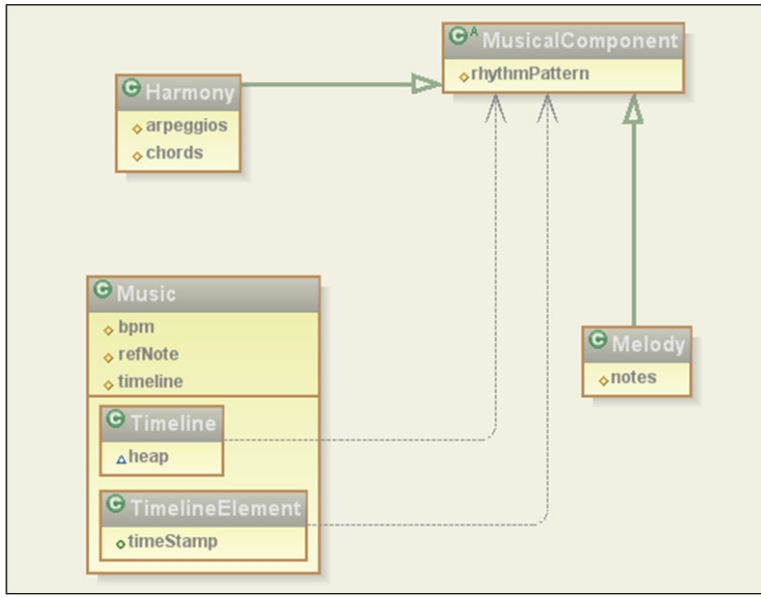
Code Example 10 shows how to code a *Harmony* composed of three chords and two different Arpeggios. Note that there is no information on the *Harmony* regarding the fingering of C*hords* (i.e. chord shapes). This knowledge belongs to the *Performer*.

Class octopus.Music

Music is normally expressed in terms of pitch (i.e. melody), rhythm (i.e tempo), and the quality of sound which includes timbre, articulation, dynamics, and texture (Dannenberg, 1993).

In the Octopus Music API, *Harmony* and *Melody* (both containing rhythmic information) are implementations of the *octopus.MusicalComponent* abstracted class.

Figure 70: Class diagram for the *MusicComponents*.

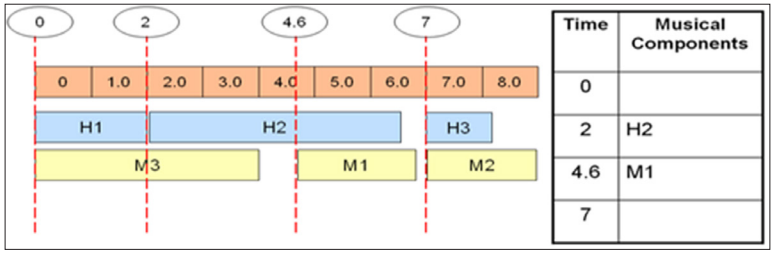


Source: Own image.

Music is a set of *MusicalComponents* scheduled in time. The timeline is represented by the *Music.Timeline* internal class, which is responsible for indexing the *MusicalComponents* throughout the duration of the *Music*. A class diagram showing the relationship between the Music and the *MusicalComponents* is presented in Figure 70.

Figure 71 shows an internal representation of *Music* composed with 3 *Harmonies* and 3 *Melodies*. Note that H1 and M3 start at the beginning of the music; hence the index is zero for both objects in the timeline. The same does not happen to H2 and M1 which was scheduled to start in a latter time requiring exclusive pointers for them in timeline table.

Figure 71: Internal musical components organisation over time.



Source: Own image.

Code Example 11: Creating a Music with harmony from a HarmonicProgression and "free notes" melody.

```
//Create and empty music object
            Music music = new Music();

//Generate the chords based on a harmonic progression of I -ii -V7 in C major.
    HarmonicProgression harmonicprogression = new HarmonicProgression("Jazz");
    harmonicprogression.addScaleDegree("I");
    harmonicprogression.addScaleDegree("ii");
    harmonicprogression.addScaleDegree("V", IntervalFactory.getMajorSeventh());
    Chord[] chords =  harmonicprogression.getChords(NoteFactory.getC());

//Create a demo RhythmPattern
   RhythmPattern rhythmPattern = RhythmPatternLibrary.getConstantRhythmPattern();
    //Assign the Chords and the RhythmPattern to the harmony
    Harmony jazzyHarmony = new Harmony(chords,rhythmPattern) ;

//Create a melody based on the freeSoloNotes array;
    String[] freeSoloNotes =  ;
    Melody melody = new Melody(freeSoloNotes,rhythmPattern);

//Insert the MusicalComponents. Harmony starts at the beginning followed by harmony.
    music.insertMusicalComponent(jazzyHarmony,0.0);
    music.insertMusicalComponent(melody,jazzyHarmony.getDuration());
```

Code Example 11 demonstrated how *Music* can be 'assembled' using a Harmonic Progression in C Major and free notes soloing. Note that the melody will only start after the Harmony has finished, with the timestamp returned by *jazzyHarmony.getDuration* method.

Musical Data Interpreters

A satisfactory realisation of an encoded work can be reconstituted through the interpretive practice of trained performers, but the knowledge that enables human performers to interpret music notation is extremely difficult to represent in a formal way (Sundberg, 1980).

Historically, the Western musical tradition has developed what we now refer to as Common Music Notation (CMN) to provide a written representation of musical compositions.

One of the key problems is that music notation is not just a mechanical transformation of performance information. Performance nuance is lost going from performance to notation, and symbolic structure is lost in the translation from notation to performance. It seems that music notation rules are made to be broken (Dannenberg, 1993) even if it was designed to serve the needs and processing abilities of humans (Loy and Abbott, 1985).

As previously mentioned, the Octopus Music API has its focus on the modelling of elements involved in a musical performance, mainly on the *Performer* and its *Instrument*. However, rather than coding each action involved in the performance, the *Performer* was programmed to interpret the Musical Structures by itself. In other words, the programmer does not have to code every minor detail as illustrated by the B.B Queen example. Instead, the Musical Data Interpreters classes are used to play the Musical Data Structures.

Mathews (1970, p.272) once said that:

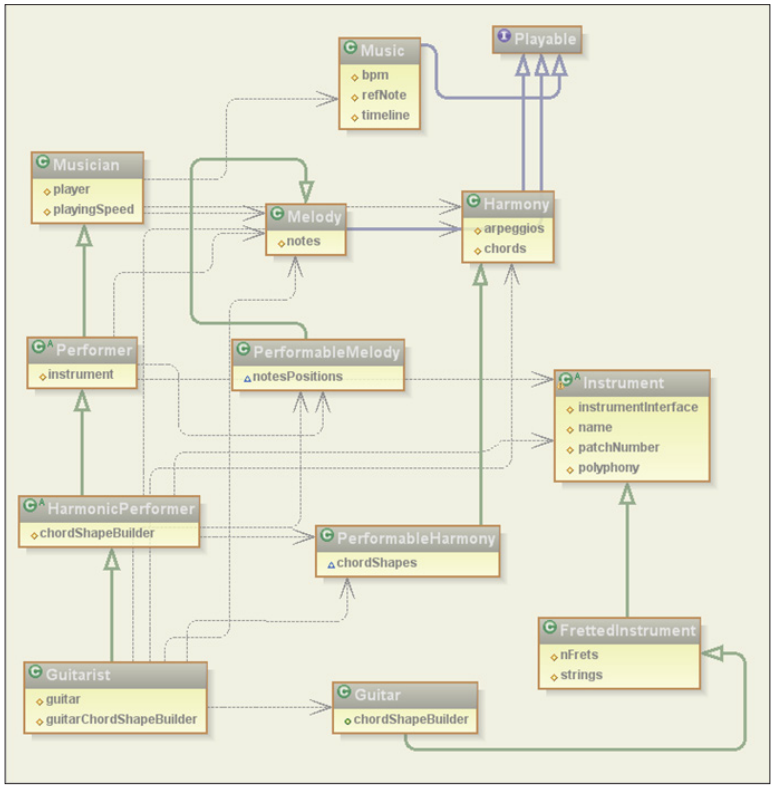
> The desired relationship between the performer and the computer is not that between a player and his instrument, but rather that between the conductor and his orchestra.

If the above quote is true, then who should be responsible to add expressivity and interpretation to a musical piece? The conductor may be responsible for directing the overall mood of the piece but surely,

he can not oversee every single action of the performers. Besides, as seen in Chapter 2, performers have their own means of interpretation.

We believe that the interpretation of the musical information varies according to the knowledge of the *Musician* and the musical context where it has been applied. A *Musician* does not need to play a musical *Instrument* to be able to understand music. In the same way, a percussionist is not less of a *Musician* for concentrating more on the rhythmic aspect of the music and less on the melody or harmony.

Figure 72: Class diagram for the Musical Data Interpreters.



Source: Own image.

Figure 72 shows the class diagram of the Musical Data Interpreters interacting with the Musical Data Structures in different levels. The first and most basic level of interpretation is the *Musician* with musical knowledge but who does not know anything about playing an instrument.

The Performer inherits all the skills of the *Musician* and implements some general monophonic instrumental knowledge. *HarmonicPerfomer* inherits all the skills of its *superclasses* and adds polyphonic instrument handling (*Harmony*). The *Guitarist* is a *HarmonicPerformer* with expertise in *Guitar* playing.

Class octopus.Musician

The *Musician* is an interpreter of the playable musical structures such as: *Scale, Melody, Harmony, Music, RhythmPattern* and so on. He knows how to read and play these structures in the simplest possible way. No instrument restriction is considered in this computation.

Class octopus.instument.Performer

The information contained in a traditional score is interpreted in performance time according to a formally incoherent set of rules, known as performance interpretation (Loy and Abbott, 1985).

A *Performer* uses the rules of musical interpretation to reconstitute an acceptable facsimile of the musical idea during the performance (Loy and Abbott, 1985). This suggests that Performers extends the knowledge of the Musicians with contextualised information about the *Instrument*.

As a subclass of *Musician*, *Performers* are also capable of interpreting musical structure, but they have to adjust these *MusicalComponents* to the characteristics of its *Instrument*. For instance, when a *Guitarist* plays a *Harmony* he will play it respecting the limitation of the specific *Guitar* that is being used, which may sound slightly different

when 'played' by the *Musician*, although a *Guitarist* is ultimately a *Musician*. This is known as polymorphism in the OOP paradigm.

All performers can play *Melody* but the same is not true in regards to the *Harmony*. A *Performer* that can play *Harmony* is represented by the class of *HarmonicPerformer*. Both *Performer* and *HarmonicPerformer* are abstract classes, so they need to be specialised for a particular *Instrument*.

As previously explained, when a *Performer* is asked to play *Music* it adjusts the musical information to its particular instrument. This 'learning' process generates enriched versions of the Harmony and Melody musical components, respectively represented by the classes *PerformableMelody* and *PerformableHarmony.* These two classes contain all the information relating to the performance of the *Music* using a particular *Instrument* such as: articulation, plucking point, chord shapes, et cetera.

Class octopus.instument.fretted.Guitarist

The *Guitarist* is a *HarmonicPerformer* that knows how to play the *Guitar.*

Since the Guitar used in the performance *can have* a direct influence in the way the *Music* is played, the *Guitarist* needs to 'know' beforehand which *Guitar* they will be playing. In OOP terms, the *Guitar* needs to be informed in the *Guitarist* constructor. Code Example 12 demonstrates a *Guitarist* being instantiated.

Code Example 12 Guitarist instantiation.

```
...
Guitar guitar = new Guitar(); //create a classical guitar
Guitarist BB_Queen = new Guitarist(guitar); //create the guitarist and assign the guitar to it

 //request the performer to show the InstrumentGraphicalInterface whislt playing.
 BB_Queen. showInstrumentLayout();

BB_Queen.play(music) //play the music
```

As a *HarmonicPerformer*, the *Guitarist* knows how to play a *Chord*. As previously stated, a *Chord* can be played in different regions of the guitar (*ChordShape*), using different fingering and *Arpeggios*. This knowledge was programmed into the Guitarist using a chord shape similarity function.

The similarity function compares the previous chord shape with the candidates' chord shapes of the next chord in the harmonic sequence. In theory, the more similar the chord shape is to the previous chord, the lower the effort to move from one to the other (travel--cost). The similarity function returns a value between 0 and 1, where 1 means the same chord (Costalonga and Miranda, 2006). The similarity functions used in this work are given by equations system bellow:

$$A = \left[a_{i,j} \right] n \times m : a_{i,j} = fSimP(PosA, PosB)$$
$$Pos = (String, Fret), NewC = [Pos], OldC = [Pos] \quad (1)$$

$$fSimP(PosA, PosB) = 1 - \frac{1}{e} \times fDis(PosA, PosB),$$
$$\forall PosA, PosB : PosA(string) = PosB(string) \quad (2)$$

$$fSimP(PosA\ \ , PosB) = \frac{1}{2} - \frac{1}{e} \times fDis(PosA, PosB),$$
$$\forall PosA, PosB : PosA(string) \neq PosB(string) \quad (3)$$

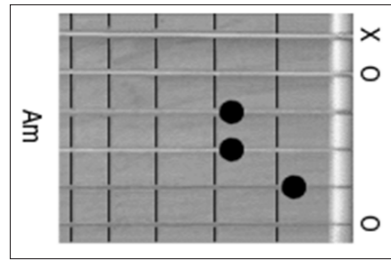
$$fDis(PosA, PosB) = \left| PosA(fret) - PosB(fret) \right|,$$
$$\forall (PosA(fret) > 0) \wedge (Pos_j(fret) > 0) \quad (4)$$

$$fDis(PosA, PosB) =$$
$$\left| \frac{\sum_{k=0}^{n-1} NewC[k][fret]}{n - qtOSN} - \frac{\sum_{k=0}^{m-1} OldC[k][fret]}{m - qtOSM} \right|,$$
$$\exists (PosA(fret) = 0) \vee (Pos_j(fret) = 0) \quad (5)$$

Equation 1 represents a matrix of *m x n* elements where m = number of positions in the chord shape and n = number of positions in the next chord shape. Similarities of each position are given by Equations (2) and (3), where e = finger span value. Equation (4) is used to calculate the distance in non-open chords whilst Equation (5) is used with open chords, where qtOS represents the number of open strings used in the chord shapes.

To exemplify, consider an Am chord shape (Figure 73) composed by the positions[string,fret] [(5,0),(4,2);(3,2);(2,1);(1,0)] which is going to be followed by a G chord. The first step is to find candidate chord shapes for the G chord. Some of them are: [(6,3);(4,0);(2,0);(6,3);(4,0);(3,4)], [(6,3);(5,2);(4,0);(5,10)], [(4,9);(1,10);(6,3);(3,4);(2,3)].

Figure 73: Am chord shape.
The black dots represent the position the fingers should be placed in. The hollow circle marks indicated the sting that must he plucked 'openly' and 'x' the string that should not.



Source: Own image.

Every candidate chord shape for the G chord will be compared with the Am chord shape, generating a matrix as shown in Table 11.

Table 11: Am to G similarity calculation.

G\A	-	(5,0)	(4,2)	(3,2)	(2,1)	(1,0)
(6,3)	-	.31				
(4,0)		.31	.62	.32		
(2,0)				.31	.62	.31

For every position of G (e.g. pos (6,3)), a value is calculated in relation to the positions of Am at the same string (Equation 3), one string above and one string below (Equation 4). For example, the position (4,0) (4th string open) of the chord G will be compared with the positions of the 5th and 3rd strings of Am chord shape, respectively position (5,0) and position (3,2).

The similarity between the chords is given by the average of the highest values of each row. In this example, the similarity is (0.31 + 0.62 + 0.62)/3 = 0.52. The candidate chords are then sorted by similarity.

The similarity is the main criteria for the selection of a chord shape but not the only one. The chord shape must also comply with the rhythmic pattern that is being used. For instance, in the case of strumming, the chord shapes with open string notes that do not belong to the basic structure of the chord must be ignored.

Obviously, the first chord shape can not be compared. Therefore, the selection is made based on the proximity of chord shape to the guitar head (the lowest fret average values).

It is important to highlight that even though the similarity function used by the current implementation of the Guitarist is backed up by the biomechanical theory travel cost proposed by Rosenbaum (1995), it is not derived from any data collected from the experiments described in chapter 4. This topic will be discussed in the second part of this Chapter.

The biomechanical study of the guitarist's right-hand (pluck hand) is not in the scope of this research. Nevertheless, the Octopus does provide two classes to allow a detailed formalisation of the right--hand techniques (arpeggios): *GuitarBar* and *GuitarArpeggio (*Code Example 13*)*, respectively extending *Bar* and *Arpeggio.*

Code Example 13: Guitar Arpeggios.

```
GuitarArpeggio gpr = new GuitarArpeggio(4);
  gpr.setBpm(240);

GuitarBar bs1 = new GuitarBar(4,4);
  bs1.addSingleRhythmEvent(bs1.WHOLE_NOTE,Bar.
          RHYTHM_EVENT_NOTE,1,0,127,
          GuitarBar.FINGERPICKING_THUMB_FINGER);
  gpr.insertBar(bs1,0);

GuitarBar bs2 = new GuitarBar(4,4);
  bs2.addSingleRhythmEvent(bs1.HALF_NOTE,1,2 ,
          bs1.DIRECTION_UP_STROKE,bs1.REGION_INDEX_FINGER,
                  bs1.FINGERPICKING_INDEX_FINGER);
  gpr.insertBar(bs2,1);

GuitarBar bs3 = new GuitarBar(4,4);
  bs3.addSingleRhythmEvent(bs1.HALF_NOTE,1,3,bs1.DIRECTION_UP_STROKE,
          bs1.REGION_MIDDLE_FINGER,
          bs1.FINGERPICKING_MIDDLE_FINGER);
  gpr.insertBar(bs3,2);

GuitarBar bs4 = new GuitarBar(4,4);
  bs4.addSingleRhythmEvent(bs1.QUARTER_NOTE,1,4,
          bs1.DIRECTION_UP_STROKE,
          bs1.REGION_RING_FINGER,
          bs1.FINGERPICKING_RING_FINGER);
  gpr.insertBar(bs4,3);
```

The slightest of the variations in the strokes of a *Guitar Arpeggio* is enough to create a whole new *Arpeggio*. Some of the stroke's properties are: direction of the stroke, fingerstyle (*PIMA*)

or pick style modes, plucking point, plectrum hardness and shape, body slap region, percussive muting, string slap intensity, and plectrum attack angle.

The programmer has the option to write the *GuitarArpeggio* himself *(*Code Example 13*)* or let the Guitarist automatically 'learn' the Arpeggio. The automatic conversion of the *Arpeggio* into *GuitarArpeggio* takes into consideration if the *Arpeggio* is meant to be strummed, arpeggiated, played with a plectrum or using the fingers. The default is the Classical Finger Style Arpeggio (PIMA).

At the moment, the *Guitarist* is the only full implementation of a *Performer* in the API but this does not mean that the Octopus API can only be used to model guitar performances. The priority for the implementation of the *Guitarist* is in accordance with the ultimate goal of our research. Nevertheless, the extension to the *Performer* class to embrace other *Instruments* would not be difficult for a Java programmer.

Instrument Classes

In the real world, musical instruments are classified by different criteria such as the note range or the way they generate the sound (what vibrates in the instrument to produce the sound). For example, in an orchestra, the instruments are split into woodwind, brass, percussion, strings.

Hornbostel-Sachs (or Sachs-Hornbostel) is a system of musical instrument classification devised by Erich Moritz von Hornbostel and Curt Sachs (von Hornbostel and Sachs, 1961), and first published in the Zeitschrift für Ethnologie in 1914. It is the most widely accepted system for classifying musical instruments by ethnomusicologists and organologists (Lysloff and Matson, 1985).

The Hornbostel-Sachs system is based on one devised in the late 19th century by Victor Mahillon, the curator of Brussels Conservatory's musical instrument collection. Mahillon's system was the first

to classify musical instruments based on what vibrated to produce its sound; however, this system was limited to western instruments used in classical music. The Hornbostel-Sachs system is an expansion on Mahillon's in which it is possible to classify any instrument from any culture.

This API adopted the way a *Performer* interacts with the *Instrument* as the classification criteria, which resembles the Sachs-Hornbostel system. However, instead of classifying the instruments based on how they produce the sound (what vibrates), we classify how the performer manipulates the instrument in order to produce the sound with it.

For instance, the ergonomics of string-fretted instruments are very similar. It does not matter whether it is an acoustic classical guitar or a mandolin. The way a performer (normally) interacts with these instruments to produce the sound is by stopping the strings against the fingerboard, consequently changing the effective length of the strings, which in turn changes the frequency at which the string vibrates when plucked.

This form of categorisation is useful in the context of this book because it favours the reuse of code. For example, the skills to play the Guitar are quite different from the skills to play the piano but it is very similar to the skills to play the mandolin since both are from the Lute family. Therefore, once an instrument of the Lute family is modelled, the rest demand little effort to be modelled.

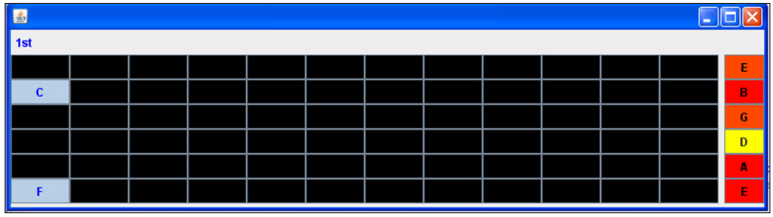
Class octopus.instument.Instrument

The abstract class *Instrument* establishes the minimum requirements that a new *Instrument* must implement to be able to interact with the other classes of API. This assures the scalability of the API to contemplate new *Instruments* during its development.

For example, all *Instruments* of the API must implement an *InstrumentGraphicalInterface* that provides visual feedback of the

performance. Figure 74 shows an implementation of such a graphical interface for the Guitar (class *GuitarGraphicalInterface*).

Figure 74: *InstrumentGraphicalInterface*: a graphical interface of the Guitar class. The 6 rows of the matrix represent the string and the columns the frets. On the right-hand side, the labels show the strings open tuning. The darker the red, harder the string has been plucked.



Source: Own image.

Class octopus.instument.string.fretted.FrettedInstrument

The *FrettedInstrument* Class represents the category of *Instruments* of the Lute family. Most plucked string instruments belong to the Lute family (such as guitar, bass guitar, mandolin, banjo, balalaika, sitar, and pipa). The lute refers to plucked string instruments with a fretted neck and a deep round back (Lysloff and Matson, 1985).

The *Guitar* class is a subclass of *FrettedInstrument* with an overridden constructor to model an acoustic classical guitar with 6 strings the standard tuning (E, A, D, G, B, E) and 12 frets clear frets.

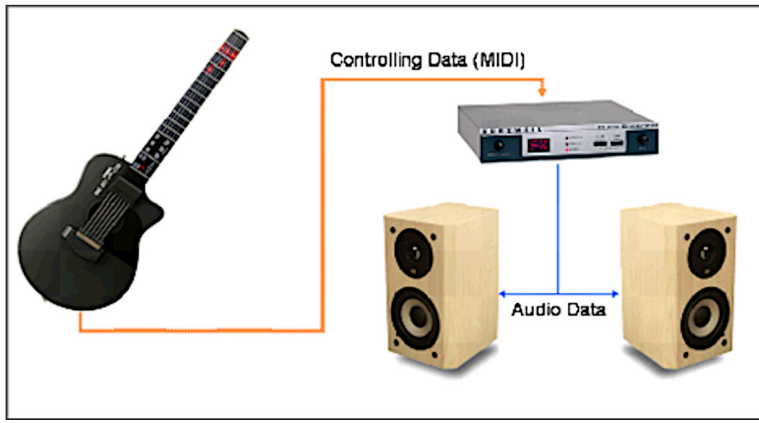
Communication Classes

Once a musical representation is adopted, issues of transmitting and storing the representation arise. Transmission, especially in real-time, raises questions of network protocols, the conventions by which information is transmitted and received. Storage raises the question of coding, or how the abstract information is converted into specific bit patterns.

MIDI is the most prevalent protocol for the real-time transmission of music information, but it has many weaknesses (Dannenberg, 1993). In a conventional MIDI setup, the controller interface (e.g. instrument) might not incorporate the Sound Generator Unit (e.g. synthesiser) itself, meaning they are two distinct devices. This separation makes the architecture more flexible once the units can be replaced to better suit a particular musical task. For example, a guitarist will most certainly find it easier to play a Guitar-shaped MIDI controller than a keyboard, which is traditionally the preferred MIDI controller.

Figure 75 illustrates a guitar-like MIDI controller sending MIDI messages to an independent external synthesiser that convert the MIDI messages into analogical wave signals that are passed to the speakers.

Figure 75: Conventional MIDI setup.



Source: Own image.

The specialisation of the units into different devices provides great flexibility but also draw attention to a fundamental element of the setup: the communication protocol that interfaces the units. The communication protocol must guarantee that the musical information generated in the controller is understood by the Sound Generation Unit, so it can render the sound accordingly.

Every musical instrument has its own means of producing sound (the mechanics of the instrument). Likewise, performers use different techniques when playing an instrument. So, how is it possible to have a standard protocol that contemplates all instruments and playing styles? It is not. The MIDI was designed for keyboard instruments and this simplification costs the expressiveness of performers of other types of instruments. In summary, MIDI sequencers will always treat musical material identically regardless of the instrument.

It is important to clarify that the Octopus API does not aim to be a synthesizer or a communication protocol; instead, it does provide the programmer with a way to communicate with external Sound Generation Units.

Normally, musical software would make direct use of programming language libraries to gain access to MIDI or audio devices. This option can be tricky since it involves low-level programming skills. Additionally, it would require the programmer to be familiar with every little detail of the Musical Data Structures classes, which goes against the concept of encapsulation of the OOP paradigm. This, however, is an inconvenience and not the main problem.

The main problem is that there is no commercial synthesizer capable of rendering expressive performances to the level of detail the Octopus API allows one to model them, as seen in Section 2.5 (p. 42). Consequently, there is no protocol either. So, what communication protocol should be used in the Octopus API to communicate with a Sound Generator Unit that does not even exist?

The solution found was the creation of a middle-tier layer that contains all the performance information but only transmits what the Sound Generation Unit can handle using whatever communication protocol it supports. This is where the *InstrumentGraphicalInterface* comes into play, providing visual feedback of what is not yet possible to sound.

Musical data can be either continuous or discrete. Continuous information changes over time and is typically represented by digital sampling, splines, or arbitrary mathematical functions. In contrast to continuous data that fills time intervals, discrete information usually represents events at a point in time (Dannenberg, 1993).

These events are usually related to the production of a sound, but the parameters or sub-actions involved in this activity vary from *Instrument* to *Instrument* and also from the capability of the synthesizer to process it.

What should a synthesizer know about a *Guitar* music performance? Is it the same knowledge necessary to render a saxophone performance? A subset of actions might be the same, but not all. How is it possible to tell the Sound Generation Unit that the force applied by the guitarist's index finger is not enough to render a clean note or that the string was plucked using a triangular plectrum in an upward movement?

Even a synthesiser specialised in a particular instrument may not be able to produce all the sound nuances that performers can produce with real instruments. Whilst the shape and the hardness of the pick could be relevant information for a particular guitar synthesiser, to another it could be pointless.

The point we are trying to make is that in the real world the performer interacts freely with a musical instrument, even in non-musical ways. Who has not heard of the famous incident of Jimi Hendrix setting fire on his guitar 1967 during a concert in Finsbury Park Astoria?

In order to provide a way to describe any possible action the instrumentalist might want to perform, the Class *MusicalEvent* extends the *java.util.Properties*. In essence, it is an unlimited collection of attributes that describes the musical actions in any level of detail that the synthesizer requires to produce the sound with fidelity. Some of the

parameters appear quite often so it was decided to make them permanent class attributes. They are: note, duration, timing and velocity

Class octopus.communication.MusicalEventSequence

Musical tasks can be described as a sequence of simple actions with specifiable goals (Pennycook, 1985). A *MusicalEvent* is no more than a single atomic musical task. An entire musical piece is likely to need more than one *MusicalEvent*.

The *MusicalEventSequence* is an array of *MusicalEvents*. It is the artefact that the *Musician* generates when requested to play something because it is the synthesizer that actually plays the *Music* through the *ShynthesizerController* class.

The *MusicalEventSequence* can also be used to group *MusicalEvents* in order to process them all together. For example, add a delay in all the events of the sequence.

Class octopus.communication.SynthesizerController

The *SynthesizerController* is a parser from the *MusicalEventSequence* to whatever protocol is used to control the synthesizer. The standard protocol supported by *Java* is still MIDI, so *MidiSynthesizerController* realises the interface SynthesizerController to parse *MusicalEvent* into MIDI messages.

The *MidiSynthesizerController* is able to communicate with external devices through the MIDI ports. In Java, this is achieved by connecting transmitters (MIDI OUT) to receivers (MIDI IN), just like you would do it with a cable.

To deal with the particularities of guitar performance, a *GuitarMidiSynthesizerController* was designed to convert guitar performance information into MIDI messages. This conversion implies a loss of precision because, as previously explained, the MIDI was not designed for guitar usage.

Justice needs to be done: MIDI would not have been the standard communication protocol for digital musical instruments for the last 20 years if it was not a well-designed solution. To overcome the lack of native support for some instruments, the MIDI specification proposes the use of System Exclusive Messages (SysEx-Messages) to extend the functionalities of the protocol to specific devices/manufactures.

The GuitarMidiSynthesizerController makes wide use of SysEx-Messages to communicate all the performance actions. However, this is only helpful if the device (receiver) could interpret these messages. Unfortunately, such a device does not exist yet.

To verify the functioning of the solution, we implemented the *GraphicalGuitarMidiReceiver* class. This class is used to decode the SysExMessages and provide visual feedback using the GuitarGraphicalInterface (Figure 74, p. 160).

MACHINE LEANING (ML)

Data analysis techniques derive either from standard statistics or computer science. While Machine Learning (ML) has been more concerned with formulating the process of generalisation, the statistics focused on testing the hypothesis (Mitchell *et al.*, 1990).

The objective of ML is to find computationally efficient solutions to data analysis problems and make intelligent decisions based on data. Like most of the AI applications, machine learning techniques have been used in non-deterministic domains that require prediction, forecasting, diagnosis, or decisions involving human judgment (Luger, 2002, p.50). Their outcome suggests that something is probable, but not necessarily true.

To select an appropriate learning algorithm, it is important to have in mind what must be 'learnt' from the data and how this knowledge will be used to draw conclusions. Witten and Frank (2002, p.38) classify the learning styles into:

Classification (Supervised) Learning: A learning scheme takes a set of classified examples from which it is expected to learn a way of classifying unseen examples. It is supervised because the success is subjectively measured by humans or objectively by comparing with a set of examples not used in the learning stage.

Association Learning: Any association between features is sought, not just ones that predict a particular class value. Association learning differs from classification learning by seeking some interesting structures in the data that could be used as an association rule to predict any attribute (even more than one), and not only the *class*. Association rules usually involve only non-numeric attributes and demand a high number of examples and high accuracy levels in the data (95% accurate).

Clustering: Groups of examples that belong together are sought. Used when classes cannot be easily identified and it seems that the elements of the group fall naturally together. The success of clustering is measured subjectively in terms of how useful the result appears to be for human use. It can be combined with classification learning to find intelligible descriptions of how new instances should be placed into the clusters.

Numeric Prediction: The outcome to be predicted is not a discrete class but a numerical quantity. A variation of the classification learning when the outcome is a numerical value rather than a category (Mitchell *et al.*, 1990). In this type of learning, the emphasis is on the importance of the attribute and how it relates to the numerical outcome.

Another aspect to consider when selecting an ML technique is the interpretability of the outcome of the learning (concept) to the human eye. The knowledge can be represented as a 'black box', whose internal mechanisms are effectively incomprehensible, or a transparent box whose construction reveals its structural patterns (Witten and Frank, 2002, p.3).

Structural patterns explicitly capture the decision structure. It can be described as one rule, a set of rules or decision tree, for instance. Not all machine learning methods produce easily understood structural descriptions. Neural Network (NN), for example, learns to classify new examples in ways that do not involve explicit structural descriptions of the knowledge that is learned. If the explanation is the main goal of the data analysis, Neural Nets is not a good choice.

Learning Strategy - Which Learning Algorithm Use?

An important objective of this research is to attempt to predict errors in guitar performances. Previously, we have discussed some of the errors that are expected to happen during a performance, such as: a) delays; b) note additions, deletions and substitutions; c) buzzed and muffled notes. These errors are believed to be correlated respectively to: speed, precision and force.

In Chapter 4 we have described two sets of experiments to measure: a) speed and precision; and b) strength and posture. The data collected is mainly numerical, which will either limit the selection to numerical prediction algorithms or will demand a pre-processing stage of *discretization*.

We have also explained that the experiments were recorded independently, which resulted in separate data sets with different attributes, each containing values with different ranges. Also, each data set has its particularities that suit a different type of algorithm.

Table 12 shows some of the characteristics of the datasets that must be observed in order to choose a suitable learning algorithm. For example, Neural Networks can handle numerical attributes, is not so susceptible to outliers (noise) but *overfits* if insufficient examples are provided.

Table 12: Strength, precision, speed, and posture data characteristics.

Data	Noisy	Numeric	Few examples	Nº Classes
Strength	✓	✓		3..4
Precision			✓	0..N
Speed		✓		1..4
Posture	✓	✓	✓	4

In ML terms, the input takes the form of concepts, instances, and attributes. Concepts are what we are trying to find. They must be intelligible and operational, so that they can be understood, represented symbolically (e.g., in terms of Octopus Music API, as discussed above) and applied to in the real world.

The instances are examples. Each instance is an individual example of the concept to be learned and it is characterised by the values of its attributes (Witten and Frank, 2002, p.38). The attribute that is to be predicted is known as the *class* of the example (not be confused by OPP class definition).

Normally, algorithms for numerical prediction work in a single class scenario, not in a multi-class; in this work, our goal is to predict not only the speed and force used by the hand during a chordshape, but the speed and force of every finger involved in the task. Hence, the data needs to be partitioned in a way that can feed several models. The selection of the attributes can be complex under these circumstances even if the models share the majority of them.

As previously explained, the hand biomechanical system is a dynamic system where the slight motion of a single finger can disturb the whole balance of the hand configuration so, ideally, the models should not be completely independent. Additionally, some learning schemes will work better in certain portions of dataset than others. Thus, choosing a single general learning scheme that will work all--round is a challenging task that demands a significant amount of trial and error.

The strategy consisting of trying several learning algorithms in a data set to select the best one is often called a toolbox approach (Freitas, 2002, p.10). We have used Weka 3.6 to assist us in performing this task.

Weka is a comprehensive tool bench for machine learning and data mining. It is an open-source project coded in Java, which provides easier integration with Octopus Music API. The advantage of using a package like Weka is that a whole range of data preparation, feature selection and data mining algorithms are integrated. This greatly facilitates the performance comparison of the different learning algorithms. Nevertheless, there are disadvantages too.

Weka does not implement the latest techniques and the documentation is quite limited. For this work, we would rather see Weka as a filter that will narrow the options of machine learning approaches that could be potentially successful with our data. Nonetheless, customisation of the algorithms might still be necessary.

The question of which is 'the best' learning scheme is very subjective. The truth is that there is no universally best learning method (Freitas, 2002, p.32). It all depends on the selection of the attribute and the descriptive power of the examples.

To demonstrate the goodness-of-the-fit of the predictions made with different algorithms, we use the same performance measures that are commonly found in the ML literature. They are:

Root Mean-Squared Error (RMSE): The Mean-Square error is the principal and most common way to quantify the difference between the actual value (a) and the true value of the quantity being predicted (p) in linear regression models. The square root is taken to give it the same dimensions of the predicted value itself.

$$RMSE = \sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}}$$

Mean Absolute Error (MAE): The Mean Absolute Error is the average of the absolute error (p –a). It is an alternative to RMSE

with the advantage of not being susceptible to outliers, once all sizes of error are treated evenly according to their magnitude.

$$MAE = \frac{|p_1 - a_1| + .. + |p_n - a_n|}{n}$$

Relative Absolute Error (RAE): Sometimes it is the relative rather than absolute error values that are of importance. For example, if a 10% error is equally important whether it is an error of 50 in a prediction of 500 or an error of 0.2 in a prediction of 2. This is our preferred measure because it allows us to compare prediction performance for all the subjects.

$$RAE = \frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + ... + (a_n - \bar{a})^2}, \text{ where } \bar{a} = \frac{1}{n}\sum_i a_i$$

Correlation Coefficient (CC): Measures the statistical correlation between the actual values (a's) and the predicted values (p's). The correlation coefficient ranges from 1 for perfectly correlated results, through to 0 when there is no correlation, to -1 when the results are perfectly correlated negatively, which should not occur for reasonable prediction methods. We use it as a complementary measure.

$$CC = \frac{S_{PA}}{\sqrt{S_P S_A}}, \text{ where } S_{PA} = \frac{\sum_i (p_i - \bar{p})(a_i - \bar{a})}{n-1}, S_P = \frac{\sum_i (p_i - \bar{p})^2}{n-1}$$

$$\text{and} \quad S_A = \frac{\sum_i (a_i - \bar{a})^2}{n-1}$$

For classification problems, we also report the percentage of correctly classified instances. All the formulae were extracted from Witten and Frank book (2002, p.148).

Machine Learning Algorithms

It is outside the scope of this research to discuss the details of the machine learning algorithms used to model speed, force or precision

data. Nevertheless, we will use this section to briefly explain some of the main ideas behind the algorithms we have used and the rationale for using them. We have mainly used three approaches: a) Instance-based (IBK); b) Decision Table; and c) Trees for numeric prediction (REPTree and MP5).

Let us start presenting the instance-based K-nearest Neighbours classifier (IBK). The nearest-neighbour method was adopted as a classification scheme in the early 1960s and has been widely used in the field of pattern recognition for almost five decades.

Fix and Hodges (1951; Silverman and Jones, 1989) performed the first analysis of the nearest-neighbour scheme, and Johns (1961) pioneered its use in classification problems. Nearest-neighbour methods gained popularity in machine learning through the work of Aha (1991), who showed that instance-based learning can be combined with noisy exemplar pruning and attribute weighting, and the resulting schemes perform well in comparison with other learning methods.

The term nearest-neighbour is normally used in statistical pattern recognition literature, whereas the term instance-based learning is typically used in the machine learning literature (Freitas, 2002, p.59). Instance-based learning however does not only include the nearest-neighbour methods but also the locally weighted regression method that assumes instances can be represented as points in Euclidean space (Freitas, 2002, p.230).

Learning for these algorithms consists of simply storing the presented training data. When a new query instance is encountered, a set of similar related instances is retrieved from memory (using a distance function) and used to classify the new query instance. This is a peculiar learning strategy in which the search proceeds from specific to general rather than from general to specific as in the case of tree or rule induction (Freitas, 2002, p.201).

Like any other learning scheme, instance-based algorithms do have some disadvantages. One of them is that performing the

selection of similar examples in real-time can be slow. This however can turn into a positive when the model needs to be expanded constantly, such as when modelling live human performance.

Another disadvantage is that instance-based algorithms can be susceptible to outliers, meaning that the examples must be good. There is an old Computer Science saying for that: "Garbage in, garbage out". No matter how intelligent a data mining algorithm is, it will fail to discover high-quality knowledge if it is applied to low--quality data (Freitas, 2002, p.201). Some algorithms can handle noisy data better than others but outliers are never good for machine learning algorithms.

Fortunately, there is a way to alleviate the impact of outliers in instance-based methods and that is exactly what IBK does; it uses several similar (neighbours) examples to classify a new instance. In our case, we have used 3KNN (K-Nearest Neighbours) because we have 3 measurements per chord during the data collection stage. The influence of the neighbours in the final prediction was weighted using a standard 1/distance function.

Indeed, the IBK performed very well with our data. A t-test comparison of the IBK with other very popular schemes for numeric prediction have shown there is a statistical significance[5] ($p < 0.05$) in the RMSE of the IBK and the others schemes when predicting the time for the 'first arrival' of the non-barre chords data for Subject 1, as can be seen in the Comparison 1. This superiority could be verified in all the arrivals, with all the chords and subjects.

5 The statistical significance is indicated in the Comparisons by the letter "v" in front of the RMSE results. The "*" indicates no statistical significance.

Comparison 1: IBK, MLP, MP5 and Linear Regression for Subject 1 Non--barre First Arrival.

```
Tester: weka.experiment.PairedCorrectedTTester
Analysing: Root_relative_squared_error
Confidence: 0.05 (two tailed)


Dataset                          (1) lazy.IBk | (2) trees (3) funct (4) funct
------------------------------------------------------------------------------
NB_t-weka.filters.unsuper(100)   11.23  |  74.16 v   93.35 v   93.88 v
------------------------------------------------------------------------------
                                 (v/ /*)  |  (1/0/0)   (1/0/0)   (1/0/0)


Key:
(1) lazy.IBk (Instance-base)
(2) trees.M5P (Model Tree)
(3) functions.MultilayerPerceptron (NN)
(4) functions.LinearRegression
```

In the first three lines of the Comparison 1 box, we can see the type of the comparison test used (T-Tester), the type variable that was analysed (RMSE), and the confidence (0.05). The bottom lines (key) show the algorithms that were compared (IBK, MP5, MLP, and LR). However, it is in the middle lines that we can see the individual RMSE per algorithm and if they are statistically significant (represented by a 'v') of not (represented by a '*'). All the comparison boxes will use the same layout.

In this particular comparison, we can see that the IBK have an RMSE of 11.23 which is statistically different from the other

results. All the comparisons use a 10 x 10 cross-validation for training the models.

Even though the results of IBK have shown some impressive potential for prediction on this particular dataset, it has its limitation. The IBK considers all attributes of the instances when attempting to retrieve similar training examples from memory. If the target concept depends on only a few of the many available attributes, then the instances that are truly most similar may well be a large distance apart (Freitas, 2002, p.231).

One approach to overcome this problem is to weigh each attribute differently when calculating the distance between two instances or, more drastically, even eliminate the least relevant attributes. However, as we increase the number of degrees of freedom available to the algorithm for redefining its distance metric in such a fashion, we also increase the risk of overfitting (Freitas, 2002, p.235).

A more sensible approach is to combine the strengths of distinctive machine learning schemes that perform well in poles apart, complementing each other. For numerical prediction, we have chosen two other learning schemes: a simple decision table majority classifier and an M5 Model Tree.

A Decision Table is one of the simplest hypothesis spaces possible, and usually, they are easy to understand. It is a precise yet compact way to model complicated logic.

Experimental results have shown that on artificial and real--world domains containing only discrete features, an algorithm inducing decision tables, can sometimes outperform state-of-the--art algorithms such as C4.5, which is often considered one of the best of its kind (Kohavi, 1995). This, however, does not mean that it will not perform well with continuous numerical data if appropriately discretized. In fact, performance is quite good on some datasets with continuous features, indicating that many datasets used in machine learning either do not require these features or that these features have few values.

Creating a decision table involves selecting some of the attributes that best represent the class. The prediction will be as good as the selected group of attributes allows it to be. The implementation of the Decision Table that we have used considers the RMSE as an evaluation measure and use Best-First (forward) to search for the 'best' subset of attributes.

In our dataset, the attributes found to be most relevant for the 'classification' of the 'time of the first arrival' (tFist) were: the fret, the vertical displacement of the index (IVD) and ring (RVD) fingers, as can be seen from the ML Example 1.

ML Example 1: Classification table on Subject 1(Non-barre/1st arrival)

Decision Table:
Number of training instances: 378
Number of Rules: 126
Non matches covered by Majority class.
Best first.
Start set: no attributes
Search direction: forward
Stale search after 5 node expansions
Total number of subsets evaluated: 31
Merit of best subset found: 7.37
Evaluation (for feature selection): CV (leave one out)
Feature set: 1,2,6,8
Rules:

Fret	IVD	RVD	tFirst
'(8.2-inf)'	'(0.4-1.3]'	'(3.1-inf)'	466.6666666666667
'(7.4-8.2]'	'(0.4-1.3]'	'(3.1-inf)'	191.33333333333334
'(6.6-7.4]'	'(0.4-1.3]'	'(3.1-inf)'	158.33333333333334
'(5.8-6.6]'	'(0.4-1.3]'	'(3.1-inf)'	177.33333333333334
'(4.2-5]'	'(0.4-1.3]'	'(3.1-inf)'	196.66666666666666
'(3.4-4.2]'	'(0.4-1.3]'	'(3.1-inf)'	174.0

...
Time taken to build model: 0.67 seconds

=== Predictions on test data ===

inst#	actual	predicted	error
1	221	219	-2
2	264	267.5	3.5
3	256	243	-13
4	179	176.5	-2.5

```
...
=== Summary ===

Correlation coefficient                  0.9965
Mean absolute error                      5.8042
Root mean squared error                  7.6928
Relative absolute error                  8.1887 %
Root relative squared error              8.298  %
```

ML Example 1 shows a simulation Decision Table. The top lines show the general parameters used to run the algorithm. The 'rules' section shows part of the rules found (126 in total) but, most importantly, it shows how the numerical data was discretized. This is followed by example 'predictions' and then the 'summary', where the performance measures are displayed. An RAE of 8.1% with a 0.99 correlation coefficient is a very good result, even better than the IBK despite being a radically different approach.

There is another learning strategy that must be considered when predicting numerical values. In reality, it should have been the first to be considered because all the attributes are numerical: the regression approach.

The classical way of dealing with continuous prediction is to write the outcome as a linear sum of the attribute numeric values with appropriate weights (regression equation). Although linear regression is an excellent scheme for numerical prediction, widely used in statistical applications, it has the disadvantage of linearity (Witten and Frank, 2002, p.114), in which case a Neural Network would be a more appropriate approach.

Neural nets are commonly used for predicting numerical quantities, although they suffer from the disadvantage that the structures they produce are opaque and cannot be used to help understand the nature of the solution. Although one can gain some insight from plotting the marginal effect of predictors, the NN inevitably introduces

complex interactions that often do not reflect reality. Furthermore, without careful control, the NN can easily overfit the data resulting in over-optimistic predictions.

A third option is the trees for a numerical prediction that mix linear regression with decision trees which make them more accurate than a simple linear regression. There are two main types of numerical trees: a regression tree and a model tree, respectively implemented in Weka by the REPTree and MP5. The main difference between a regression and model tree is that the latter finds one regression equation per whilst the former use just one for the whole tree .

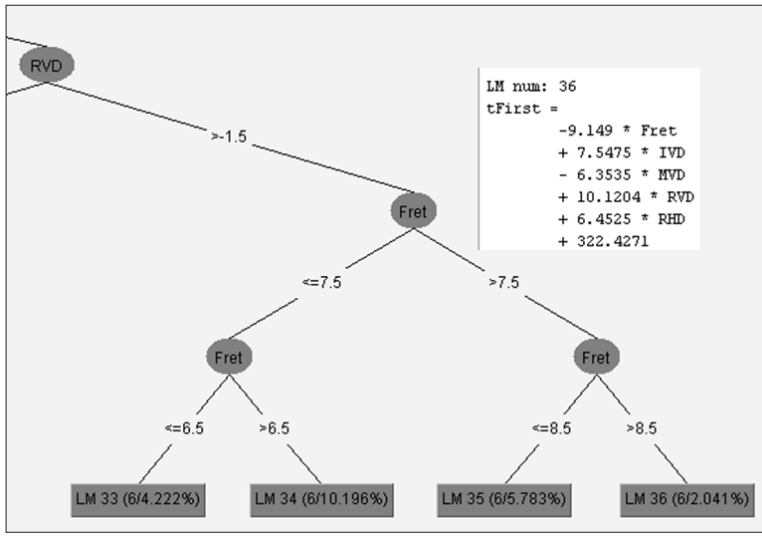
All three approaches were tried and although none have proven to be particularly effective for the speed dataset, we believe that the suitability of regression approach for numerical prediction can not be ignored. A comparison (Comparison 2) between these three regression approaches has shown that the regression (REP) and model tree (MP5) perform better than the simple linear regression and Neural Network Multilayer Perceptron (MLP).

Comparison 2: Numeric Trees, Linear Regression, NN MLP comparison.

```
Tester: weka.experiment.PairedCorrectedTTester
Analysing: Root_relative_squared_error
Datasets: 1
Resultsets: 4
Confidence: 0.05 (two tailed)


Dataset                   (2)REPTree | (1) MP5    (3) LR   (4) MLP
----------------------------------------------------------------------------------
NB_t-weka.filters.unsuper(100)   66.95   |  74.16   93.88 v   93.35 v
----------------------------------------------------------------------------------
                          (v/ /*)   |   (0/1/0)   (1/0/0)    (1/0/0)


Key:
(1) trees.M5P (Model Tree)
(2) trees.REPTree (Regression Tree)
(3) functions.LinearRegression
(4) functions.MultilayerPerceptron (Neural Net)
```

The RMSE between the two types of numeric prediction trees was not statistically significant at 66.95 and 74.15 RSME. We have therefore opted to use the MP5 model tree because it performed slightly better overall and produced smaller trees.

Figure 76: Part of the MP5 Model Tree for Subject 1(Non-barre/1sr Arrival)



Source: Own image.

Figure 76 shows a branch of a tree constructed with MP5 algorithms for Subject 1 (Non-barre/1st arrival) and one of the linear models (LM) used to predict the speed value of that particular leaf. For example, the Linear Model 36 (LM36) only applies for a scenario in which the ring finger is not moving downwards over two strings (RVD > -1.5) and the Fret > 8.5.

Experience has shown that combining the predictions from multiple methods often yields more accurate and robust predictions than can be derived from any one method (Witten and Frank, 2002). To do so, we need to use a meta-learning scheme.

We have used two different meta-learning schemes: Stacking (Seewald, 2002) and Bagging/Voting (Kuncheva, 2004), respectively applied to numerical and classification uses.

In stacking, the predictions from distinct classifiers are used as input into a meta-learner, which attempts to combine the predictions to create a final best-predicted classification. Hence, in our proposed

solution, the used predictions from three classifiers (IBK, MP5 and Decision Table) as input variables into a Linear Regression meta--classifier, which attempts to 'learn' from the data how to combine the predictions from the different models to yield maximum classification accuracy.

The concept of bagging (voting) is commonly used to address the inherent instability of results when applying complex models to relatively small data sets (Statsoft, 2009). As we have discussed, learning schemes will perform differently for different dataset. In small data sets, this variation can be extremely high, raising the recurring question of which learning scheme to use. Bagging (voting) solves this issue democratically: voting, as the name says.

The final classification is the one most often predicted by the different schemes, which in our case are: Random Forest (Breiman, 2001), Random Committee/Random Tree (Dietterich, 2000), and IBK.

Both Random Forest and Random Committee are already meta--schemes by themselves and use their own strategy to select the best classifiers. Their classification strategy will not be discussed because the decision to use them was purely based on performance rather than on their suitability to the data. The Bagging (voting) meta-learning scheme only performs well if all the inner-schemes also perform well. Hence, we selected the three that performed best.

A Note on Data Preparation

Real data is often of disappointingly low quality (Witten and Frank, 2002, p.48). Preparing input for data mining investigation usually consumes the bulk of the effort invested in the entire data mining process. According to Cabena, Stadler et al (1998), data preparation accounts for up to 60% of the effort in data mining.

In the following sessions, we will present how the Speed, Precision, Posture and Force data were prepared to suit different

learning schemes. However, there is one key difference worthy of mentioning first:

In Section 4.3 (p. 106) we explained that strength can be measured under static (isotonic) or dynamic conditions (isokinetic). In our experiments, we have measured them under static conditions, in other words, the subjects were not really performing during the measurement but rather applying pressure in a shape of a chord over a measuring tool that resembles a guitar. For this reason, the examples used to model force and postures are merely descriptors of chord shapes. Any other kinetic attribute that may impact force (i.e. speed) is not used in the example.

Contrastingly, speed and precision were recorded under dynamic conditions. The subjects had to move from the reference position to the established chord shape position, therefore it is the movements and not the chord shapes that are used as examples in this scenario.

More details of movement and chord shape descriptors will be provided in context throughout the next sections.

Modelling Chord Speed

Given two chord shapes, how long does it take to move from one to another? If this time is greater than specified in the music score then a delay is likely to occur.

Delays, like any other error, can be caused either by cognitive (e.g. timekeeping mechanisms discussed in section 2.3.4, p.34) or biomechanical (e.g. muscle speed discussed in section 3.2.3.3, p. 79) constraints. Whatever the source of the delay, it seems that individual delays are not cumulative as demonstrated Heijink and Meulenbroek (2002). They found that guitarists tend to speed up to compensate for an anticipated time loss caused by an upcoming complex task to be, on average, on time.

Identifying the transitional time required to go from one chord shape to another is just one aspect of learning. The other aspect refers to the order of the fingers arriving at their respective place.

The fingers' arrival order is especially relevant if chords are being strummed; releasing a vibrating string prematurely will generate noise (*pick/pull off*). Likewise, placing a finger on a string that is already vibrating may not be enough to dampen the sound entirely, transferring part of the vibration intensity to the new note (*hammer-on*).

The selection of a learning scheme suitable to model the speed of all three subjects has proven to be difficult. No one algorithm will model equally well the data from all three subjects because they have distinct performing styles. While Subject 2 produced very 'homogenous' data, Subject 3 performed the chords more erratically.

The more homogenous (not to be confused with pattern regularity) the data is the more difficult it is to find patterns because one concept overlaps the other.

In order to assess the predictive power of the data, we used an Expectation Maximisation (EM) clustering algorithm. EM assigns a probability distribution to each instance, which indicates the probability of it belonging to one of the clusters. We inputted the algorithm with the expected number of clusters (number of non-barre chords = 7) in advance, but no class was specified (as usual in a clustering algorithm).

Table 13 shows the not so impressive results of the EM clustering algorithm in attempting to classify part of the instances of non-barre chords (from the bottom reference only) based solely on the fret and speed of each digit (index, middle, ring). Approximately, only 30% of the instances are correctly classified. This means that data does not fall naturally into classes. This highlights the importance of data preparation.

Table 13: EM clustering speed results for non-barre chords.

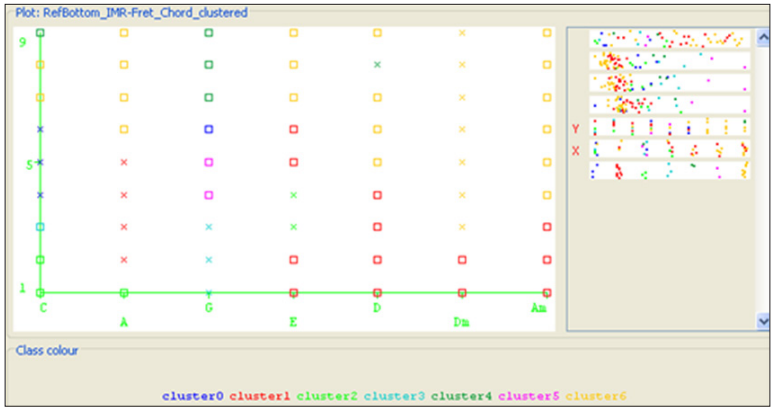
A classification table (per subject) is presented to indicate the classes of the misclassified examples. For example, take chord C for Subject 1; the row 'C' and column 'C' show that 3 classifications were correct but other instances of C chord were misclassified as E (2x), G, D, and Dm (2x).

Subject 1 - Incorrectly clustered instances :					**43.0**	**68.254 %**	
	C	A	E	G	D	Am	Dm
C	3	0	2	1	1	0	2
A	0	4	1	0	0	0	4
G	1	0	0	3	3	2	0
E	0	4	2	0	0	0	3
D	0	4	0	0	1	0	4
Dm	0	2	0	0	0	0	7
Am	0	3	0	0	0	0	6
Totals	4 (6%)	17 (27%)	5 (8%)	4 (6%)	5 (8%)	2 (3%)	26 (41%)
Subject 2 - Incorrectly clustered instances :					**44.0**	**69.8413 %**	
	Dm	A	C	D	E	G	A
C	2	0	3	4	0	0	0
A	0	2	0	3	1	0	3
G	0	2	1	3	0	2	1
E	1	0	1	1	3	1	2
D	0	2	2	4	0	0	1
Dm	2	0	2	2	1	1	1
Am	0	0	1	2	3	0	3
Totals	5 (8%)	6 (10%)	10 (16%)	19 (30%)	8 (13%)	4 (6%)	11 (17%)
Subject 3 - Incorrectly clustered instances :					**46.0**	**73.0159 %**	
	Am	A	E	G	C	Dm	D
C	0	1	3	0	2	3	0
A	0	2	4	0	2	0	1
G	1	1	0	1	0	4	2
E	0	1	4	0	2	0	2
D	0	2	3	0	2	0	2

Dm	1	0	1	0	0	5	2
Am	1	2	4	0	1	0	1
Totals	3 (5%)	9 (14%)	19 (30%)	1 (2%)	9 (14%)	12 (19%)	10 (16%)

In theory, the better the classification made by a clustering algorithm, the more suitable the data is to Machine Learning, therefore requiring less effort in the data-preparation stage. Table 13 shows not only the percentage of correct classification considering the chord as the class but also a classification table to help analyse the classification of the chords individually. For example, take chord C for Subject 1; row 'C' and column 'C' show that 3 classifications were correct but other instances of C chord were misclassified as E (2x), G, D, and Dm *(2x).*

Figure 77: Clustering classification of speed for Subject 1.
The x-coordinates of the graph show the clusters (chords) and the y-coordinates show the fret position.



Source: Own image.

Figure 77 shows a graphical way of presenting the classification table of clustering algorithms. Note that, for Subject 1, only the chords C (around 5th fret) and A (around 1st and 5th frets) presented a pattern. This indicates that this piece of data, as it is, is not highly

suitable for machine learning, although supervised learning schemes may present better results.

In order to improve the learning rate, the input must be honed to make it more amenable to learning schemes; attribute selection, discretization, and data cleansing are examples of these procedures. Another strategy is to create new synthetic attributes in order to present existing information in a form that is suitable for the machine learning scheme.

If a linear relationship involving two attributes A and B is suspected, and the algorithm is only capable of axis-parallel splits, the ratio A/B might have to be defined as a new attribute (Witten and Frank, 2002, p.231). With the speed data, rather than just providing the departure and arrival chord shapes, we must indicate the relationship between departure and arrival positions and we do that by defining the motion involved in this translation.

The movements are described by the vertical (VD) and horizontal (HD) displacement of the fingers that are used to perform the chord shapes. To calculate the VD and HD, the string and fret values of each position of a chord shape is subtracted from the previous chord shape position that is being performed by the same finger. For example, a move of the index finger from the position (string, fret) = (3,3) to position(1,5) would result in a VD= -2 and HD = 2.

Table 14 shows the vertical (VD) and horizontal (HD) displacement values per finger (I, M, R) of the non-barre chords, which have used only the index, middle and ring fingers.

Table 14: Vertical and horizontal displacement of the fingers for non-barre chords. Ref. is the frame of reference, VD = vertical displacement, and HD = horizontal displacement. I, M, and R represent respectively the finger Index, Middle and Ring. tMax is the time required to perform de chord shape.

Chord	Ref.	Fret	IVD	IHD	MVD	MHD	RVD	RHD	tMax
C	Bottom	9	1	0	2.55	0	4	0	522
C	Top	1	-4	0	-2.3	0	-1	0	441

A	Bottom	9	3	1	1.7	0	1	-1	281
A	Top	1	-2	1	-3.45	0	-4	-1	206
G	Bottom	9	4	1	4.25	1	0	0	591
G	Top	1	-1	1	0	1	-5	0	426
E	Bottom	9	2	0	3.4	0	3	-1	237
E	Top	1	-3	0	-1.15	0	-2	-1	457
D	Bottom	9	2	1	0	0	1	0	351
D	Top	1	-3	1	-5.75	0	-4	0	461
Dm	Bottom	9	0	0	1.7	0	1	0	180
Dm	Top	1	-5	0	-3.45	0	-4	0	493
Am	Bottom	9	1	0	2.55	0	2	-1	195
Am	Top	1	-4	0	-2.3	0	-3	-1	468
...									

Based on the average finger size of the adult male population (Pheasant and Haslegrave, 2006, p.144) the middle and little finger fingers were weighted slightly differently from the index and ring finger, which was weighted = 1. The middle finger is more suitable for reaching higher strings (bass) than low strings, so was weighted 0.85 for upward movement (less effort) and 1.15 for downward movement (more effort). The little finger was the opposite, being highly weighted in upward movements (1.25) and less so in (0.75) the opposite direction.

In the last column of Table 14 the tMax class is shown. tMax is the time that is necessary to perform the described movement, which corresponds to the time of the slowest finger. Three fingers are used to play non-barre-chords hence there are also three 'arrivals' per instance, tMax being the third arrival.

Table 15 shows a sample of the input data that is presented to the learning schemes in order to predict the speed of movement. Each instance (row) shows the same movement performed in different frets and its respective arrival times.

Table 15: Sample of the input data used to train the model for 'arrivals' in non-barre chords.

I, M, and R represent the finger Index, Middle and Ring respectively. VD = vertical displacement, HD = horizontal displacement. tFirst, tSecond, and tThird are the times of the first, second, and third 'arrivals'

Fret	IVD	IHD	MVD	MHD	RVD	RHD	tFirst	tSecond	tThird
1	1	0	2.55	0	4	0	287	446	463
2	1	0	2.55	0	4	0	168	373	385
3	1	0	2.55	0	4	0	471	510	558
...									

As previously stated VD and HD stand respectively for vertical and horizontal displacements. I, M, and R stand for the index, middle and ring fingers. So, for example, IVD is the index vertical displacement. tFirst, tSecond, and tThird are the time of the arrivals; no information regarding the finger order of arrival is presented.

In theory, some learning algorithms allow a multi-class prediction. In practice, Weka does not implement this feature. Therefore, every arrival had to be modelled independently.

The fingers' arrival order shares the same attributes as the speed of the arrivals, but it requires a different learning strategy because the class is nominal. Most of the classification algorithms perform better with nominal classes then numerical classes. In fact, most of the learning schemes implemented in Weka only support nominal classes. This dramatically increases the number of algorithms that can be used, and consequently the chance of finding one that can perform better.

Table 16 shows sample input data used to train a model to predict the finger's arrival order. Once again, it is presented as a multi-class situation. However, instead of training the model independently as we have with the numeric data, we can simply concatenate the attributes into a single class. For example, the first instance of Table 16 has a class 'RMI'.

Table 16: Sample of the input data used to train the model for fingers' arrival order in non-barre chords.
I, M, and R represent the finger, Index, Middle and Ring respectively. VD = vertical displacement, HD = horizontal displacement. First, Second and Third indicated the finger related to the first, second and third 'arrivals'.

Fret	IVD	IHD	MVD	MHD	RVD	RHD	First	Second	Third
1	1	0	2.55	0	4	0	R	M	I
2	1	0	2.55	0	4	0	R	I	M
3	1	0	2.55	0	4	0	R	M	I
...									

A note must be made regarding the classification of the finger's arrival order. In Chapter 4 (p. 96), we saw that Subject 2 was much more regular in the way he placed his fingers on the freetboard. This regularity can be easily spotted by the classification algorithms which led to a classification rate of 100% accuracy. The same cannot be said from Subject 1 and 3 in which the classification did not perform so well, 70% and 63% of positive classification respectively. However, this is still well above the 'random' classification threshold.

To improve the classification rate, a filter that removes the misclassified instances according to the J48 Classification Tree was applied before the final model was generated. The filter removes approximately 20-25% of the data that is considered to be outliers, considerably improving the performance rate of classification schemes. The results presented in the next section refer to this filtered dataset.

As previously explained in Chapter 3 (p. 70), barre and non--barre chords present significant biomechanical differences. While the former uses a palmar pinch grip the later uses a tip-pinch grip. Anatomically, they also require a different set of muscle and different levels of voluntary muscle contraction (MVC).

These differences, however, do not necessarily require separate data preparation or the use of different learning algorithms.

However, the barre technique demands that one finger (usually index) presses several positions at the same time, therefore is not possible to calculate the movement distance in the same way as the non-barre chords. In order to describe a barre technique too many attributes would be necessary for the input dataset. This would lead to a problem known as the curse of dimensionality.

Curse of dimensionality (Bellman, 1961) refers to the exponential growth of hypervolume as a function of dimensionality. In practice, the curse of dimensionality causes learning algorithms with lots of (irrelevant) inputs to behave badly.

To avoid the 'curse', barre-chords were processed separately from the non-barre chords. In actual fact, not one but two modelling approaches were used.

The first approach is very similar to the one used with the non-barre-chords; the difference is that the distance (VD and HD) are not calculated to the index finger. Instead, two new attributes believed to influence the overall speed of the barre--chords were added:

1. nStr – number of strings the barre covers; nStr = 6 for the F Chord and nStr=5 for B and Bm chords.
2. HHM – Hand Horizontal Motion: HHM = 0 for the F chord and HHM = 1 to B and Bm chords

In addition, little finger displacement is now also included in the input. Table 17 shows a sample of the data used to model the speed of the fingers (except the index) in a barre-chord situation.

Table 17: Sample of the input data used to train the model for 'arrivals' in barre-chords.
I, M, and R represent the finger, Index, Middle and Ring respectively. VD = vertical displacement, HD = horizontal displacement. t1, t2, t3 are the times of the first, second, and third 'arrivals'.

Fret	nStr	HHM	MVD	MHD	RVD	RHD	LVD	LHD	t1	t2	t3
8	6	0	-3.45	0	-1	0	-1.5	-1	129	208	287
9	6	0	-3.45	0	-1	0	-1.5	-1	189	236	318
1	5	1	2.55	2	2	1	1.25	0	541	579	619
...											

The second model is just for the barre technique itself. Modelling the behaviour of the barre independently from the other fingers helps to reduce the number of attributes, avoiding the curse of dimensionality as previously explained. However, this is not the sole advantage of this method.

Due to the mechanics of the barre technique the strings tend to be pressed almost at the same time starting either from the top or from the bottom (never from the middle). The time gap between the notes is very small and in order to find the speed of the middle positions of the barre more attributes had to be used to better describe the barre itself; these attribute would be irrelevant for the other finger predictions.

Table 18: Sample of the input data used to train the model for the barre technique. nStr is for the number of strings the barre covers, nN is the number of notes the barre produces, VD is the vertical distance from the previous finger position to the top string of the barre, HHM is the hand horizontal motion, N is the string where barre meets a note, and tPN is the speed for N.

Fret	nStr	nN	VD	HHM	MVD	MVD	MHD	RVD	RHD	LVD	LHD	N	tPN
1	6	3	5	0	2	1.7	0	4	0	3.75	-1	6	433
2	6	3	5	0	2	1.7	0	4	0	3.75	-1	1	353
3	6	3	5	0	2	1.7	0	4	0	3.75	-1	1	441
...													

Table 18 shows the attributes used to model the barre; as an example, take the first instance (row): it indicates that the 6th string of F chord shape coming from the bottom reference took 433 ms to be pressed.

Speed Results

The results were calculated using 10-fold cross-validation. The data was not normalised, meaning that the RMSE is only used as a comparison reference between the learning schemes for the same subject.

To measure numerical prediction, RAE and Correlation Coefficient (CC) are suggested as better measures of the models. Our success benchmark is RAE < 15% with a CC > 0.95. For the classification model, we aimed to archive a positive classification superior of 95%.

Table 19 present the results of non-barre chords for all three subjects. The '1st', '2nd' and '3rd' columns refer to the 'arrivals', mentioned earlier. They were calculated using the stacking meta-learning scheme (IBK, MP5, and Decision Table) for numerical prediction. The 'Order' column refers to the arrival order of the finger (i.e. IMR, RMI, etc.) and was calculated with the bagging (voting) meta-learning scheme (Random Forest, Random Committee, IBK) for classification.

Table 19: Learning performance of the Non-barre chord models.
1st, 2nd, 3rd refer to the predicted times for the first, second, and third 'arrivals'. The Order column refers to the classification of the finger's arrival order.

Subject 1	**1st**	**2nd**	**3rd**	**Order**
Correlation	0.996	0.9963	0.995	NA
MAE	6.3022	8.8959	11.8451	0.0464
RMSE	8.2778	12.0086	17.4824	0.126
RAE	8.89%	8.00%	8.45%	18.55%
Positive Class.	NA	NA	NA	97.10%
Subject 2	**1st**	**2nd**	**3rd**	**Order**
Correlation	0.997	0.996	0.9966	NA
MAE	3.2817	4.891	6.2852	0.0255
RMSE	4.6955	6.011	8.0996	0.0748
RAE	9.12%	13.26%	9.67%	13.54%
Positive Class.	NA	NA	NA	100%
Subject 3	**1st**	**2nd**	**3rd**	**Order**
Correlation	0.9938	0.9914	0.9897	NA
MAE	6.5743	7.7043	8.7596	0.0486
RMSE	8.6975	10.3227	11.6943	0.1284
RAE	10.81%	12.37%	13.09%	19.25%
Positive Class.	NA	NA	NA	96.27%

Analysing the results in Table 19 we can conclude that the learning schemes combination worked effectively for all three subjects. Interestingly, the smaller RAE (best fit) came from Subject 1, who was found to be the slowest and least refined skill.

Regarding the fingers' arrival order, the best results are for Subject 2. Subject 1 and 3 had the input filtered (removing misclassified instances according to a J48 classification tree) to produce acceptable results and, even after that could not match the 100% positive classification of Subject 2 data. This confirms that Subject 2 was very regular in this strategy of finger placement.

Table 20 present the results of the barre-chords for all three subjects. The '1st', '2nd' and '3rd' columns refer to the arrivals of the non-barre fingers, which will invariably be the middle, ring or little fingers (not necessarily in this order). The prediction result for the order of arrival is presented at the 'Order' column.

We have assumed that the barre is performed by the index finger so the speed at which the index finger arrives in a certain position of the barre is represented by 'tPN' column, calculated also using the stacking meta-learning scheme (IBK, MP5, Decision Table).

Table 20: Learning performance of the barre-chords models
1st, 2nd, 3rd refer to the predicted times for the first, second, and third 'arrival'. tPN is the time of arrival of the positions within the barre. The Order column refers to the classification of the finger's arrival order.

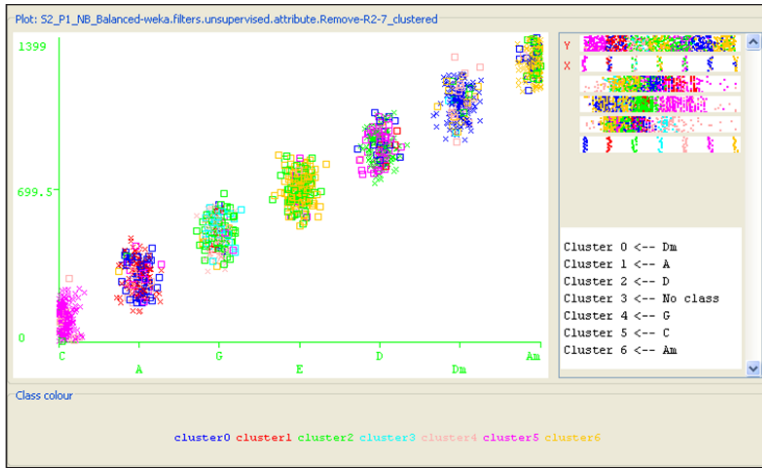
Subject 1	**1st**	**2nd**	**3rd**	**Order**	**tPN**
Correlation	0.997	0.9964	0.993	NA	0.9954
MAE	7.4186	9.5964	12.6926	0.0105	11.1743
RMSE	10.1203	12.0281	15.8999	0.036	21.881
RAE	7.26%	8.76%	12.11%	4.74%	5.96%
Positive Class.	NA	NA	NA	100%	NA
Subject 2	**1st**	**2nd**	**3rd**	**Order**	**tPN**
Correlation	0.997	0.996	0.9966	NA	0.9639
MAE	3.2817	4.891	6.2852	0.0255	11.7343
RMSE	4.6955	6.011	8.0996	0.0748	40.7629
RAE	9.12%	13.26%	9.67%	13.54%	10.02%
Positive Class.	NA	NA	NA	100.00%	NA
Subject 3	**1st**	**2nd**	**3rd**	**Order**	**tPN**
Correlation	0.9938	0.9914	0.9897	NA	0.997
MAE	6.5743	7.7043	8.7596	0.0486	6.8391
RMSE	8.6975	10.3227	11.6943	0.1284	11.8305
RAE	11.11%	12.37%	13.09%	19.25%	5.64%
Positive Class.	NA	NA	NA	96.27%	NA

Once again, Subject 1 was the easiest to predict arrival time for, and Subject 2 the easiest to predict fingers' arrival order for. The positions of the barre technique presented a slightly inferior result for Subject 2 but still very reasonable at 10% RAE. Overall, it seems that the more regular and skilful the guitarist, the more difficult is to predict speed data.

Modelling Force and Posture Data

At first sight, the force data appears to be more suitable for machine learning schemes compared to the speed data. Running the same EM Clustering Algorithm previously applied to the speed data it is possible to observe that the instances fall more naturally into categories.

Figure 78: EM Clustering analysis of non-barre chords at the 1st fret for Subject 2. The x-coordinates of the graph shows the clusters (chords) and the y-coordinates represent an internal variable used by the algorithm.



Source: Own image.

Figure 78 presents a snapshot of EM building the clusters. It is possible to observe the instances of some chords very well delineated, like the C (cluster 5). In fact, classification algorithms are capable of finding the class of a chord with a success rate of over 97% using only the force values produced by each finger (Comparison 3). This indicates that the data indeed holds some correlation to the force produced by the finger and the chord shape.

Comparison 3: Chord classification by force production.
Comparison between three classification algorithms (Bayes Net, NBTree, and J48) using the force values per digit as input and chord as the class.

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   1
Resultsets: 3
Confidence: 0.05 (two tailed)


Dataset                   (1) BayesNet | (2) NBTree (3)J48
--------------------------------------------------------------------------------
'S2_Force_NB_P1-weka.filt(100)   97.76 |   97.48 *   91.42 *
--------------------------------------------------------------------------------
                                 v/ /*) |   (0/0/1)   (0/0/1)


Key:
(1) bayes.BayesNet
(2) trees.NBTree '
(3) trees.J48
```

Regrettably, discovering the chord shape based on the finger's force values is the opposite of what we are trying to achieve. We

aim to find the force of each finger when performing a certain chord shape. To accomplish that, we need to find the attributes that best describe the chord shape.

One possible way to describe a chord shape is by its fingering. Ironically, this does not necessarily mean that the finger must be indicated. The chord shape description could be simply a list of positions (string, fret) that needs to be pressed in order to perform the chord. For example, the C chord shape can be written as [(5,3); (4,2); (2,1)].

Table 21 shows the fingerings for the non-barre chords in the first region of the fretboard (fret 1..4).

Table 21: Attributes of Non-barre chords.
'IS' stands for the Index finger in a particular String position, 'IF' for the Index finger in a particular Fret position. The same applies to the middle (M) and ring (R) fingers.

ChordShape	IS	IF	MS	MF	RS	RF
C	2	1	4	3	5	3
A	3	3	2	2	2	2
G	5	2	6	3	1	3
D	3	2	1	2	2	3
E	3	1	5	2	4	2
Dm	1	1	3	2	2	3
Am	2	1	4	2	3	2

There is another way to describe the chord shape that is less trivial: the upper limb posture. As explained in Chapter 4, certain 'frozen positions' (FPs) were extracted from the position data and these FP can function as chord shape predictors. It is assumed that the posture data have a more direct correlation with force production as seen in Chapter 3 (p. 74). Table 22 lists the most common FPs per chord and fret.

Table 22: Frozen Positions (FP).

The values are the MIDI readings related to the angle of the articulation recorded by the Exo-Skeleton.

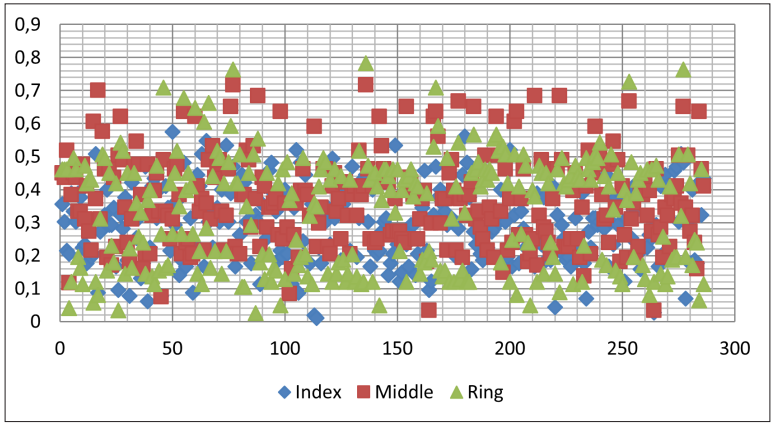
Chord	Position	Wrist	Forearm	Elbow	Shoulder
C	P1	36	12	59	74
C	P5	38	12	55	69
C	P9	41	18	52	63
A	P1	51	21	69	71
A	P5	49	25	62	69
A	P9	46	34	55	61
G	P1	33	21	59	75
G	P5	33	28	52	72
G	P9	36	38	49	66
E	P1	49	8	65	74
E	P5	46	12	59	67
E	P9	51	15	52	60
D	P1	41	25	44	74
D	P5	43	25	41	67
D	P9	46	38	47	61
Dm	P1	43	21	72	73
Dm	P5	46	18	65	68
Dm	P9	46	28	65	61
Am	P1	38	15	78	75
Am	P5	36	15	65	69
Am	P9	33	21	65	63
Bm	P1	25	12	62	72
Bm	P5	28	15	59	70
Bm	P9	30	21	55	65
F	P1	28	12	69	75
F	P5	30	15	62	71
F	P9	33	21	59	64
B	P1	17	8	59	72

B	P5	17	12	55	69
B	P9	22	25	49	65

What initially appeared to be a fairly easy fit for machine learning algorithms now appears to be far more complex. The reason is that even the smallest variation in posture impacts the recorded force. Unfortunately, the equipment used to capture the data did not have any synchronisation mechanism between force and posture to let us capitalise on this variance. Instead, we end up with part of the force readings with a range too wide for a machine learning algorithm to make sense of it. The best results achieved did not exceed 60%. RAE.

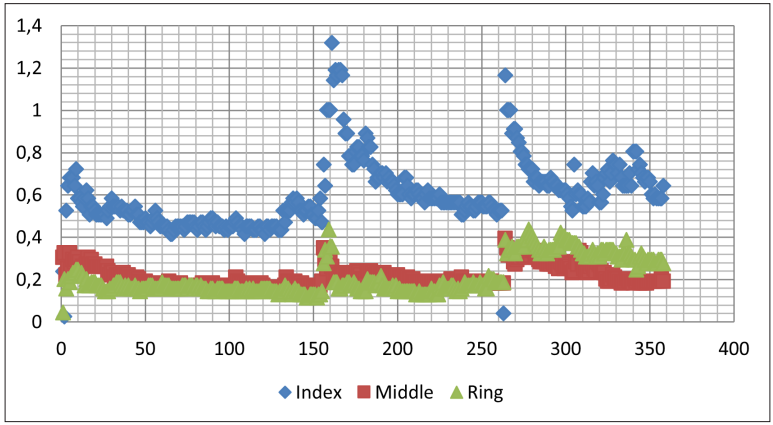
Figure 79 and Figure 80 show two very different force reading examples. While the finger' force values for the Dm chord is very regular and easily approachable by an ML algorithm, the G seems to be chaotic.

Figure 79: Force reading – G (Fret 1) Subject 2;
The x-coordinates are the number of readings and the y-coordinates the force measured.



Source: Own image.

Figure 80: Force reading - Dm (Fret 5), Subject 2.
The x-coordinates are the number of readings and the y-coordinates the force measured.

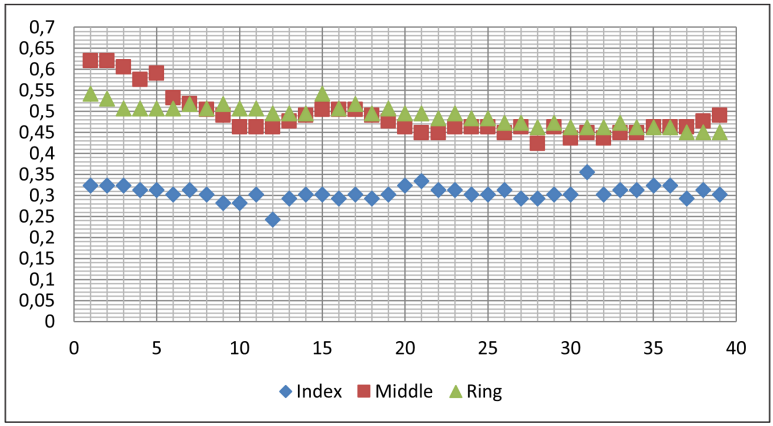


Source: Own image.

ML algorithms perform poorly with frenzied data such as shown in Figure 79, so in order to increase the learning rate we had to pre-process the data and extract the best examples to represent the chord shape. This was done by analysing the frequency distribution of the readings of every chord per fret and selecting the 10 and 30 most frequent examples respectively for barre and non-barre chords. The difference in the number of selected examples between barre and non-barre chords is proportional to the number sensors used in readings: 9 for barre-chords (3 fingers + 6 for the barre) and 3 for non-barre (3 fingers).

Figure 81 shows an example of the post-processed data extracted from the G chord data shown in Figure 79.

Figure 81: Cleaned force reading – G (fret 1), Subject 2.
The x-coordinates represent the sequential order of the reading and the y-coordinates the force measured.



Source: Own image.

The training stage of the force modelling is not very different from that for speed modelling, with one main difference. As previously explained, the force data were recorded under static conditions so instead of modelling the 'movement', we modelled the hand static position which could use the fingering and/or the FP as descriptors.

The algorithms that best performed for the force data were similar to the three used for the speed data numerical prediction, however instead of using an MP5 (model tree), we decided to use REPtree because this time the difference in performance was statistically significant.

Using a stacking meta-leaning scheme composed of REPTree, Decision Table and the IBK classifiers, we ran a comparison (Comparison 4) to determine the relevance of the posture and the fingering descriptors in the prediction of the force. The options analysed were: a) posture and fingering; b) fingering only; and c) posture only.

Comparison 4: Posture relevance for force prediction.

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Root_relative_squared_error
Datasets:   3
Resultsets: 1
Confidence: 0.05 (two tailed)


Dataset                                   (1) Stacking
----------------------------------------------------------
a) S2_P1-5-9_Fing_and_Posture  (100)  15.10
b) S2_P1-5-9_Fingering_only    (100)  15.12
c) S2_P1-5-9_Posture_Only      (100)  15.07
----------------------------------------------------------
(v/ /*)



Key:
(1) meta.Stacking (REPTree, Decision Table and the IBK)
```

As can be seen in Comparison 4, all three options performed very similarly with RMSE around 15 grams. Against our expectations, the dataset containing the posture and fingering did not produce any statistical gain. To make this worse, it considerably slowed down the training stage.

While the fingering data is a direct translation from the spatial domain and does not require any complex algorithm to be found, the posture is not as easy to be extracted from the chord shape, requiring independent models for every upper limb articulation.

For the reasons above, we have decided not to use the posture data at this stage to predict the finger's force distribution. Nevertheless, we still believe the analysis of posture data can strongly

contribute to the improvement of educational methods of guitar training and the ergonomics of guitar manufacturing.

Table 23 shows part of the input data used to train the force model for the index finger. The force is given by the *fIndex* and the other attributes are spatial coordinates of the finger position (same attributes of Table 21). The *Chord* column is just an illustrative reference and it is not used in the actual training.

Table 23: Sample of the input data used to train the index finger force model for Non-barre chords.
fIndex is the force of the index finger and the other attributes are spatial coordinates of the finger position where 'IS' stands for the Index finger in a particular String position, 'IF' for the Index finger in a particular Fret position. The same applies to the middle (M) and ring (R) fingers.

Chord	Fret	IS	IF	MS	MF	RS	RF	fIndex
A	1	4	2	3	2	2	2	0.088136
A	5	4	2	3	2	2	2	0.020339
A	9	4	2	3	2	2	2	0.00339
Am	1	2	1	4	2	3	2	0.155387
Am	5	2	1	4	2	3	2	0.155387
Am	9	2	1	4	2	3	2	0.073189

The preparation of the data for the barre-chords is more time consuming than non-barre chords. For the reasons discussed in the speed data modelling, the barre (index finger) is modelled separately from the middle, ring and little fingers.

Due to the limited number of examples of barre-chords, it is necessary to be careful when selecting the descriptors. Nonetheless, we believe that the position of the other fingers in relation to the barre has a direct impact on the force the guitarist is able to apply on the barre.

Table 24 shows an example of the attributes used to describe the barre of a B chord. They are:

1. nStr.: number of strings the barre covers or the upper string the barre reaches.
2. nN: Number of notes the barre produces;
3. pStr.: Even if just a couple of notes need to be produced by the barre, we have opted to model all 6 strings the barre may eventually cover. pStr. is the string the force (fPN) is related to.
4. MS, MF, RS, RF, LS, LF: String and fret coordinates respectively for the middle, ring and little fingers;
5. fPN: Force applied in a particular string (pString) of the barre.

Table 24: Sample of the input data used to train the index finger force model for barre-chord (the barre technique).

Fret	nStr.	nN	pStr.	MS	MF	RS	RF	LS	LF	fPN
1	5	2	1	4	2	3	2	2	2	0.204
1	5	2	2	4	2	3	2	2	2	0
1	5	2	3	4	2	3	2	2	2	0
1	5	2	4	4	2	3	2	2	2	0.009
1	5	2	5	4	2	3	2	2	2	0.188
1	5	2	6	4	2	3	2	2	2	0.040

The middle, ring and little finger models use the same attributes as the barre itself, shown in Table 24 However the *fPN* was replaced by the force of a particular finger. In addition, there is no need for *pString* since the finger in a tip-pinch grip should only press one string at a time.

Force Results

The models were trained using 10 fold cross-validation with 630 instances for non-barre and 90 for barre-chords. Similarly, to the speed, the RAE and CC were used to measure success.

Due to the initial technical problems with the equipment reported in Chapter 4 (p. 108), we were expecting a less accurate prediction than was achieved in the speed modelling. Ideally, we wanted to maintain the Relative Absolute Error (RAE) at approximately 20% with a confidence interval over 0.95. Unfortunately, despite our best efforts to highlight all the patterns in the pre-processing stage this was not possible at all times. Nevertheless, an RAE < 30% can still be considered a reasonable result.

Table 25 present the results of non-barre chords for all three subjects. Each column shows the results for the finger's force model. It should be remembered that the non-barre chords did not use the little finger and the chord fingering was the same to all subjects.

Table 25: Learning performance of the Non-barre chords force models.

Subject 1	**Index**	**Middle**	**Ring**
Correlation	0.973	0.9765	0.9647
MAE	0.0145	0.0112	0.0109
RMSE	0.0191	0.0145	0.0149
RAE	21.4227 %	20.6907 %	25.5004 %
Subject 2	**Index**	**Middle**	**Ring**
Correlation	0.9885	0.985	0.9848
MAE	0.0121	0.0172	0.0151
RMSE	0.0162	0.0239	0.0194
RAE	13.5158 %	14.4991 %	18.1679 %
Subject 3	**Index**	**Middle**	**Ring**
Correlation	0.9746	0.9743	0.9697
MAE	0.0129	0.0107	0.0083
RMSE	0.0163	0.0137	0.0107
RAE	20.283 %	23.8152 %	22.0704 %

Table 26 present the results for the barre-chords. The evaluation of the results needs to be made in a case by case basis as generalisations per subject are not possible. For instance, while the models

for Subject 2 non-barre chords seem to provide the best fit for the data among all the subjects, they also present the worst result for the barre (fPN) model.

Table 26: Learning performance of the barre-chords force models.

Subject 1	fPN	Middle	Ring	Little
Correlation	0.9866	0.9908	0.9596	0.9516
MAE	0.01	0.006	0.0053	0.0048
RMSE	0.0169	0.0086	0.0074	0.0073
RAE	11.4787 %	10.9011 %	23.336 %	23.2545 %
Subject 2				
Correlation	0.9576	0.9964	0.9754	0.9566
MAE	0.0131	0.0083	0.0049	0.0057
RMSE	0.0227	7.6606 %	0.0063	0.0073
RAE	20.7787 %	9.6549 %	20.0571 %	29.3128 %
Subject 3				
Correlation	0.9899	0.986	0.954	0.9649
MAE	0.0066	0.0068	0.0031	0.0061
RMSE	0.0143	0.0087	0.005	0.0091
RAE	8.0771 %	16.15 %	19.8382 %	21.0144 %

In the barre-chords models the relatively high RAE from the ring and little finger compared with middle finger attracted our attention. This demonstrates that the force values recorded for these fingers were very sparse, which could have two causes: a) they are highly subjective to the force from the index and middle fingers; or b) they did not hit the sensors properly.

Modelling Precision Data

Unlike the speed and force data, the precision data has very few examples; therefore, precision errors are difficult to predict using ML algorithms.

As seen in Chapter 4 (p. 96), the subject with the highest number of errors (slips) had 22 errors out of 999 opportunities, 2.2% HEP. Subject 1 had an error rate of only 0.2% HEP. Such small error rates could be easily linked to equipment failure or any other factor that is not guitarist/guitar related. The error likely had a random unknown cause in which no pattern can be detected.

With such a small number of error examples, every single one counts. They need to be individually analysed to have their relevance assessed. In case they are proven to be more than a random error, we need to examine the story behind them. What can we learn from error? What is the reoccurrence probability? These are the questions that need to be answered. However, unlike humans, computers are not good at learning with just one error.

Another aspect of error that was not sufficiently explored in the experiment is the correction strategy. When a guitarist hits the wrong position, what happens? Does he try to correct it? If so, how long does that take?

Palmer and Van de Sande (1993) shed some light on the topic reporting that musicians (pianists) tend to ignore the error and carry on with the performance. An explanation for that might be related to motor limitations. The fact is, in order to engage in error correction a musician must continuously monitor their activity by hearing the outcome of its performance. Reaction to auditory stimuli is about 30 to 50 ms (Thompson et al., 2006). Once the error stimulus has been detected by the performer (and probably by the listener too), a physical action must take place to attempt to correct the error. The time it takes to complete the movement component of a response depends on the nature of the movement, but a minimum of 300 ms can be expected (Sanders and McCormick, 1993, p.219), which is probably too late.

To make matters worse, a transition between chord shapes (movement) can present zero or several errors per finger and position, each with a different cause and yet interrelated. If we make an analogy with auto-sport racing, we can compare the guitarist with a driver that

attacks a turn wrongly and ends up with the vehicle in a track position that will not let him perform the next few turns correctly. The same happens with guitarists, one error may trigger a chain of errors.

It is unlikely that so many correlations can be found in such a small dataset. Perhaps, the 'randomness' aspects of errors make them virtually impossible to be predicted at all. Fortunately, we might not have to 'predict' the errors in order to achieve our main goal.

Let us remind ourselves of the objective of such modelling: to create computational models capable of simulating a guitar performance that resembles a human player. Unlike the ergonomic, we are not interested in finding the cause of the errors and how they could impact productivity or possibly harm the guitarist. Here, we are interested in 'recreating' the errors and not necessarily 'predict' them, which would demand a far greater database.

As seen in the previous section, instance-based learning is the only machine learning scheme capable of drawing generalisations from specific examples and we believe it is the most suitable approach to model errors. In addition, instance-based learning can continually accumulate examples (knowledge) postponing the decision-making process.

Weka does implement several algorithms for instance-based learning; The IBK in particular has been successfully used to model both speed and force. The same approach with precision data is not feasible due to the low number of examples and would certainly generate an over-fitted model.

An over-fitted model is not a bad thing if a prediction is not the goal. It is an effective way to create highly specialised rules applicable to one subject in a particular situation. This however may not work because the subject does not make the same error every single time the same scenario is recalled.

We need to be able to determine the odds of an error happening in a particular circumstance. For that reason we have decided to develop our own instance-based algorithm designed specifically to model guitar errors.

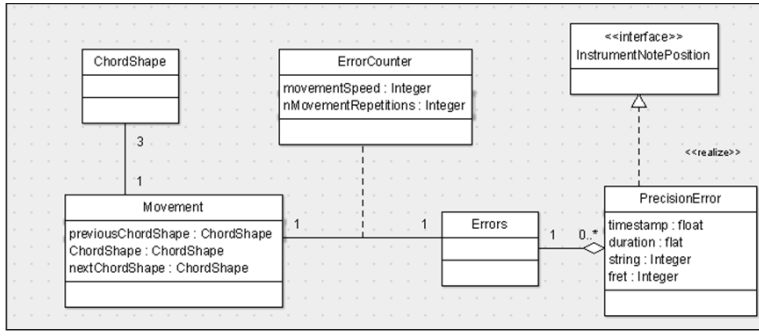
Like speed, precision errors are linked with movements, which are described by vertical and horizontal displacements calculated for every chord transition.

The algorithm we have developed adopts a 'closed world' assumption, meaning that only positive examples are modelled and the rest are assumed to be negative. In this case, the error occurrence is a positive example. Hence, any movement that was not perfectly executed is stored in the database. The number of times the movement is performed is also stored so that the probability of error can be calculated.

Since speed impacts precision, the counting takes place within adjustable speed ranges according to the error frequency distribution.

The properties of the error itself include the moment it happened (timestamp), its duration, and its absolute position (string, fret). Errors can be recorded in isolation or within a group but they are always considered a single entity when replicated. Figure 81 shows the class diagram (OOP) for precision error instance-based algorithm.

Figure 82: Class Diagram of the precision error KNN algorithm.

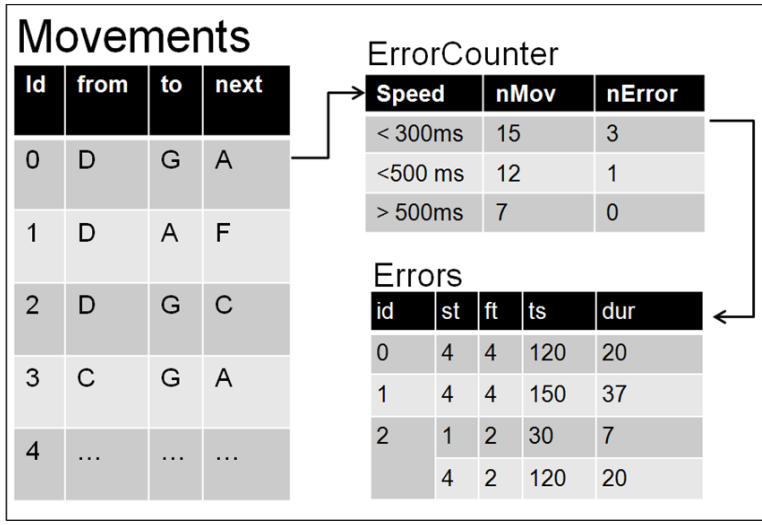


Source: Own image.

As can be seen in Figure 81, the *Movement* is not just characterised by the vertical and horizontal displacements from the previous

to the actual chord shape. The next chord shape in the sequence is also recorded for comparison purposes, as later explained.

Figure 83: Data used in the precision error modelling example.



Source: Own image.

Figure 83 illustrates an example of the algorithm data structure. The 'Error Counter' is an array used to count the number of movements (nMov) and the number of errors (nError) in a particular speed range (Speed). The 'Errors' array stores the string (st), fret (ft), timestamp (ts), and duration (dur) of the error.

For example, consider the movement from the chord shape D to G followed by A; it was executed 34 times (15 + 12 +7) and the highest number of errors occurred when the transition from D to G took less than 300 ms.

In the 'less than 300 ms' range, three errors were recorded. The first two errors (id 0 and 1) were actually two instances of the same error; therefore, it seems to be a frequent occurrence (higher probability). The third error (id 2) is a chain of errors.

To establish whether to add an error or not to a computer--generated guitar performance, the algorithm performs a probability analysis. For instance, consider a performance that contains the same movement from D to G followed by A. The first step is to find out the speed in which the movement is performed (speed model). Let us assume the movement takes 420 ms.

The value found is greater than the 300 but less than 500 ms. The total number of repetitions in the range 'less than 500' is 27 (i.e. 15 + 12) and just one error was recorded in the range between 300 – 500 ms. This gives us initial odds of 1 in 27 (3.7%). The algorithm attempts to introduce precision errors at this rate 3.7% but the moment when the error occurs is randomly selected as well as which error to simulate. Naturally, the errors with more entries have a higher probability of selection.

A 3.7% chance of error is a reasonable rate for a 'humanised' music performance. In fact, with such a small database the probability that an error will be recreated is very slim. On the other hand, such over-fitted models can also add too many of the same errors. This is where the K-Nearest-Neighbour (KNN) concept comes into play.

So far, the example described used 1 KNN, in other words, it was looking for exactly the same movement used in the training stage (D to G followed by A). In the database there are two other movements from D to G (ids 1 and 2). One followed by F and the other followed by C. If the KNN = 3, these other two records will also contribute to the error probability rate.

Let us assume odds of 1 in 13 (7.6%) and 2 in 22 (9.09%) respectively for the movement 'D to G followed by F' (id 0) and 'D to G followed by C' (id 1). The algorithm can be configured to work with:

1. The average probability: The errors of the all 3-Nearest Neighbours movements, in the particular speed range, are candidates to be included 6.7% ((7.6 + 9.09 +3.7)/3) of the times the original movement happen;

2. The highest probability: The errors of the movement with the highest probability (movement id 1) are included 9.09% of the times that the original movement is made;
3. The lowest probability: The same reasoning as the previous option (ii), but using the movement with the lowest probability (3.7%);
4. Weighted probability based on a distance function: Using a distance function to calculate similarities in the movements in order to use the errors associated with them. This is the default option and also the most elaborate one, as explained in the sequence.

The distance function is the selection criteria to find the nearest neighbours. It is a simple Euclidean distance measure from the fingers' vertical and horizontal displacement. It is composed of two parts: a) the main chord shape transition (i.e. D to G) weighted at 0.6; and b) the next chord shape transition (i.e. G to F or G to C) weighted 0.4.

The fingering used to calculate the vertical and horizontal displacement (movement) is given in Table 27. 'S' stands for string 'F' for fret; 'I', 'M', 'R', 'L' stands respectively for the index, middle, ring and little fingers. Note that the F chord is the only one that makes use of the little finger.

Table 27: Fingering of the chords used in the precision modelling example. 'IS' stands for the Index finger in a particular String position, 'IF' for the Index finger in a particular Fret position. The same applies to the middle (M), ring (R) fingers, and little (L) fingers.

Chord	IS	IF	MS	MF	RS	RF	LS	LF
C	2	1	4	2	5	3	-	-
A	4	2	3	2	2	2	-	-
G	5	2	6	3	1	3	-	-
D	3	2	1	2	2	3	-	-
F	6	1	3	2	5	3	4	3

The method to calculate the vertical and horizontal displacement was already explained in the Section 5.2.4 (p. 177), however two additional points can be observed regarding the transition to and from barre-chords (e.g. mov(G,F)).

The first point is related to the displacement for the index finger. The positioning of the index finger is assumed to be the one in the highest string (bass-string) of the barre. In the case of the F chord shape, it is the position (6,1).

The second point is the procedure to calculate the displacement of a finger that was not being used - e.g. mov(G,F). In this situation the calculation is done based on the location of the adjacent finger, preferably the one on the right (palm facing – i.e. the ring finger would be the one on the right of the middle finger). Table 28 show the finger's vertical and horizontal displacement values for the primary and secondary transitions used in the example illustrated in Figure 83.

Table 28: Calculated values for the primary chord transition (precision modelling example)

I, M, R, and L represents the finger, Index Middle, Ring and Little respectively. VD = vertical displacement, HD = horizontal displacement.

Translation	IVD	IHD	MVD	MHD	RVD	RHD	LVD	LHD
mov(D, G)	2	0	5	1	-1	0	-	-
mov(G, A)	-1	0	-3	-1	1	-1	-	-
mov(G ,C)	-3	-1	-2	-1	4	0	-	-
mov(G, F)	1	-1	-3	-1	4	0	**3**	**0**
mov(C,G)	3	1	2	1	-4	0	-	-

The distance between movements is given by Equation 5. It is simply the average of the absolute distance between the original movement and the nearest-neighbour candidate movement, where '*m1*' is the original movement, and 'm2' is the candidate movement. '*vd*' and '*hd*' is the vertical and horizontal displacement of each finger; '*nfingers*' is the number of fingers involved in the movement.

Equation 5: Movement distance equation.

$$dist(m1,m2) = 1 - \frac{\sum\left(\left|vd(m2) - vd(m1)\right| + \left|hd(m2) - hd(m1)\right|\right)}{nfingers \times 2}$$

In our example, we need to find the likelihood of the secondary transition from the original movement to the KNN candidate movements, represented by '*sm1 = dist (mov(G,A), mov(G,C))'* and '*sm2= dist(mov(G,A), mov(G,F)'*. Bear in mind the primary moment is the same for both options: mov(G,D);

Table 29 shows the distance values of the movements. When a movement is equal to another its distance is 1. The secondary movement that is the most similar the mov(G,A) is the mov(G, C) with a distance of 0.33.

Table 29: Calculate distances for the secondary transition of precision modelling example.

I, M, R, and L represents the finger, Index, Middle, Ring and Little respectively. VD = vertical displacement, HD = horizontal displacement. Dist(x) is the calculated distance between the movements.

Movement Distance	IVD	IHD	MVD	MHD	RVD	RHD	LVD	LHD	Dist(x)
dist(mov(G,A),mov(G,A))	0	0	0	0	0	0	-	-	1
dist(mov(G,A),mov(G,C))	2	1	1	0	3	1	-	-	0.33
dist(mov(G,A),mov(G,F))	2	1	0	0	3	1	3	0	0.25

To calculate the final distance between the movements, the weights must be applied to the primary and secondary movements. Table 30 lists the weighted distances for all the movements in the database.

Table 30: Final weighted distances of the movements used in precision modelling example.
Distance calculated to the movements shown in Figure 83.

id	to	from	next	Distance
0	D	G	A	1
1	D	G	C	0.73
2	D	G	F	0.7
3	C	G	A	0.6

From the movements illustrated in Figure 82 , the 'D to G followed by C' would be the 2nd Nearest-Neighbour with a distance of 0.73 (1 x 0.6 + 0.4 x 0.33). Going back to the probability calculation, this would mean that the probability of using the errors of the movements 'D to G followed by C' will be greater than using the errors of the movement 'D to G followed by F'.

The number of nearest-neighbours can also be determined by a similarity threshold. In the previous example, a similarity threshold of 0.7 would be adequate. The higher the similarity threshold the more specialised is the error and vice-versa. The similarity threshold is a good way to control the number of errors inserted into the generated performance.

EQUIPPING THE OCTOPUS API WITH BIOMECHANICAL-INSPIRED MODELS

In the previous section we have explained how the speed, force and precision data were independently modelled using ML schemes. In this section we explain how they have been put together to simulate a guitar-performance.

In the first part of this chapter we briefly presented the Octopus Music API, a Java library designed to model music performances. We have seen that Guitarist is a class that encapsulates all the knowledge necessary to play musical material (playable interface) in the Guitar.

Neither the Guitarist nor the Guitar was equipped with the attributes designed to take advantage of these biomechanical-inspired models. The focus was to implement a *Musician* that could adapt the music performance based on the instrument it was playing. Now, we want to go a step further into this specialisation and simulate the impact of the constraints of the human body in this adaptation, mainly focusing on the production of errors.

In order to make use of the new models we have extended the *Guitarist* and *Guitar* classes with its respective counterparts *IdiomaticGuitarist* and *IdiomaticGuitar*, which is explained as follows.

Class octopus.idiomatic.IdiomaticGuitar

The *IdiomaticGuitar* extends the Guitar class by specifying detailed mechanical attributes, mainly regard to guitar dimensions and string gauge.

At the current stage, the guitar body dimensions do not play an active role in the way the Guitarist precision model interacts with the Guitar but it is likely to do so in the future, hence we already included a slot for this in the model. The attributes modelled for the guitar body include: body length, upper width, lower width, and depth (Figure 7, p. 52).

Even though dimensions are not considered in the calculation of precision errors, they are in force prediction. For the latter, the most relevant attributes for the *IdiomaticGuitar* are: scale length, number of clear frets, and string data (tuning, diameter, tension). Usually, string manufacturers provide the string diameter and tension data; however, the tension value is calculated using a scale length and tuning that might not be the same in the guitar used. Therefore, the *IdiomaticGuitar* also implements methods to adjust the string's manufacturer tension to its own dimensions and tuning.

Code Example 14 demonstrates how to model an *IdiomaticGuitar*. The first step is to model the strings which equip the guitar.

In our example, six strings were modelled using the data provided by the manufacturer (D'Addario Model EJ27N). The strings, the number of clear frets, and the guitar scale length are informed at the moment the guitar is created; in this case, we called the guitar 'Admira Concerto'.

Another important attribute to calculate the force required to produce a clean note is the string action (per fret). The last few lines of the code shown in Code Example 14 demonstrates a simplified way to define the string action. The string action can be set manually, through a [String, Fret] matrix of scale action values (mm), or automatically as seen in the example; In order to set the string action automatically a reference string action value must be passed together with a constant step value that is added or subtracted from the reference based on the fret location. In the example, the scale length for the 12^{th} fret is 4 mm and the step value is 0.250 mm, then the scale length for 11^{th} fret is 3.75 mm (Table 4, p. 55).

Code Example 14: Defining an Idiomatic Guitar.

```
// Model for a set of D'Addario Classic Nylon Strings
// Material: Silverplated Wound and clear nylon
// Gauge - Normal Tension;
GuitarString[] daddarioClassicalNylonStrings = {
// 1st string, tuned in E5, string diameter = 0.71,...
//...tension informed in by the manufacturer = 6.94 kg,...
//... tension calculated on the reference scale length = 648.
            new GuitarString(1, NoteFactory.getNote("E",5),0.71,6.94,648),
            new GuitarString(1, NoteFactory.getNote("B",3),0.82,5.26,648),
            new GuitarString(1, NoteFactory.getNote("G",3),1.02,5.49,648),
            new GuitarString(1, NoteFactory.getNote("D",3),0.74,7.08,648),
            new GuitarString(1, NoteFactory.getNote("A",2),0.89,6.80,648),
            new GuitarString(1, NoteFactory.getNote("E",2),1.09,6.35,648)
};
//Create the Admira Concerto Guitar with 12 clear frets, D'Addario Strings,
// and scale length of 650 mm.
IdiomaticGuitar admiradConcerto = new IdiomaticGuitar (12,
                                        daddarioClassicalNylonStrings, 650);
//Automatically populate the sting action values;
// arg1: String action on the 12th fret = 4mm;
// arg2: Decrement of string action for the previous fret = 0.125; e.g. 11th
fret = 3.75 mm;
    admiraGrandConcerto.calculateStringAction(12,4, 0.250);
```

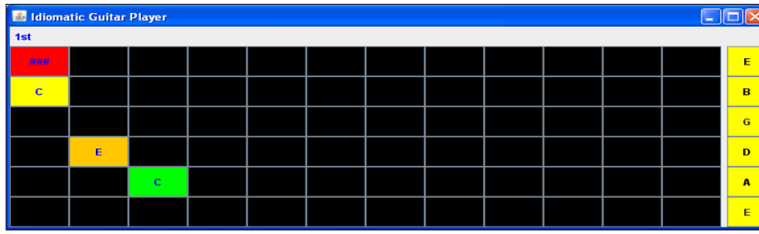
The string tension and action are the two parameters that are used to verify if the predicted force is enough to produce a clear note. The quality of the note varies according to the predicted force the *IdiomatiGuitarist* 'virtually' applies to depress the strings. At this stage, three categories of sound quality were implemented:

1. Muffled note: Less than 75% (inclusive) of the necessary force to produce a clear note; this value is proposed based on extracted force measurement discussed in Section 3.1.6 (p. 56).
2. Buzzed note; More than 75% but less than enough pressure to produce a clear note.
3. Clear note: Enough force to fully depress the string against the fretboard in a particular fret-region.

Naturally, this list can be extended and modified by the programmer according to the capabilities of the Sound Generation Unit used.

As an extended version (subclass) of the Guitar, the *IdiomaticGuitar* must also implement a visual representation of the performance. This was done by extending the *GuitarGraphicalInterface with a colour scheme that* shows when a position was accidentally pressed (precision) or if it was pressed without enough force. Figure 84 shows the visual representation of an idiomatic performance.

Figure 84: IdiomaticGuitarGraphicalInterface class.
The position in red and labelled with '###' represents a precision error, the positions in yellow represent a muffled note, orange a buzzed-note, and green a clean note.



Source: Own image.

Class octopus.idiomatic.IdiomaticGuitarist

The class *IdiomaticGuitarist* summarises everything that has been discussed up to this point. It merges the theoretical guitar playing knowledge with the limitations of the human body, represented

in the form of the biomechanical-inspired models of speed, precision and force.

As we previously explained, the guitarist must adapt his performance style not only to the guitar but also the musical style. In fact, we have mentioned other situations that can potentially impact the performance but were not fully studied in the scope of this book, such as: MPA, low temperature, injuries or a training state. In summary, the performance is susceptible to many other factors that were not touched upon by this research, which was limited to the force, precision and speed the left-hand digits. For this reason, the *IdiomaticGuitarist* is equipped with a vector of biomechanical-inspired models; one for each scenario of aggregate circumstances that could impact performance. These scenarios can be saved and retrieved as one wishes.

In the Octopus Music API, all background processes required to play a Musical Data Structure (such as *Music*) take place immediately before the execution of a piece of music. This means that all the decisions regarding chord shape selection, speed, precision errors and force are calculated and stored in form of a performable *Music* before it can actually be played. We have called this stage the 'learning' stage; it is at this stage that the Musical Data Interpreter classes (e.g. *IdiomaticGuitarist*) actually take all the decisions regarding performance actions. This stage should not be confused with the 'learning' stage of the Machine Learning Algorithms.

In order to instantiate an *IdiomaticGuitarist object*, four attributes must be informed: the location of the pre-generated force models, the location for pre-generated speed and fingers' arrival order models, the precision errors training file, and the *IdiomaticGuitar*.

The models of force, speed, and "fingers' arrival order" are loaded and used through the imported Weka classes. A discussion on the technical matters of the integration of Octopus and Weka is beyond the scope of this book because this is purely a software engineering task, which should be straightforward for any Java programmer. For more details see Weka (2009).

The precision error model, however, was implemented directly on the *IdiomaticGuitarist*, which calculates the probability of error based on the training data file passed to the constructor, as explained in Section 5.2.6.1 (p. 202).

Code Example 15 demonstrates how to instantiate and request the *IdiomaticGuitarist* to perform a harmonic sequence. The *HarmonicProgression* in question is a usual Flamenco harmonic sequence (Fernández and Rodemann, 2005). From the HarmonicProgression, we draw out the chords in the key of E Major – four chords in total. These four chords are inserted into the *Harmony* line and are played using a pre-defined *GuitarArpeggio*. As explained in Section 5.1.2.9 (p. 148), the timing in which each chord is played is defined by the RhythmPattern associated with the Harmony.

Code Example 15: BB_Queen

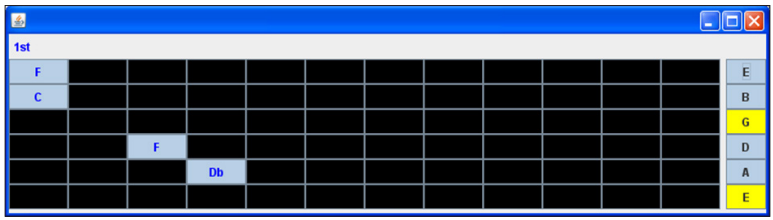
```
//CREATING THE GUITARIST
IdiomaticGuitarist BB_Queen = new IdiomaticGuitarist(admiraGrandConcerto,
                                "../forceModels", "../SpeedModels", "../Errors.txt");

//DEFINING A HARMONIC SEQUENCE
HarmonicProgression flamencoChords = new HarmonicProgression("Flamenco Cadence");

        flamencoChords.addScaleDegree("II", IntervalFactory.getMajorSeventh());
        flamencoChords.addScaleDegree("VI", IntervalFactory.getMajorSeventh());
        flamencoChords.addScaleDegree("II", IntervalFactory.getMajorSeventh());
        flamencoChords.addScaleDegree("I" );

//GETTING THE CHORDS FROM THE HARMONIC SEQUENCE
 Chord[] chords = flamencoChords.getChords(NoteFactory.getE());

//SETTING UP THE HARMONY
Harmony harmony = new Harmony(RhythmPattern.getDemoRhythmPattern());
harmony.addChord(chords, GuitarArpeggio.getDemoGuitarArpeggio());

//PLAYING
BB_Queen.showInstrumentLayout();
BB_Queen.play(harmony);
```

The first decision the *IdiomaticGuitarist* tries to make when requested to play a *Harmony* regards which chord shapes to use in order to play the *Chords*, just as the normal implementation of the *Guitarist* would do it. Even though the process of selecting of the chord shape is not the focus of the current research, we have used the *IdiomaticGuitarist* chord shape suggestion to exemplify the workflow involved in a guitar music performance simulation. The proposed chord shapes are presented by Figure 85, Figure 86, and Figure 87.

Figure 85: First and third chord shapes of the harmonic progression (degree II7).

Source: Own image.

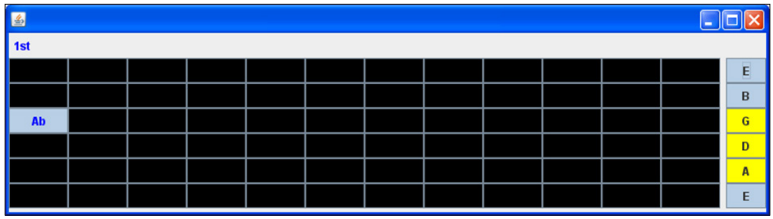
Figure 86: Second chord shape of the harmonic progression (degree VI7).



Source: Own image.

As shown by Figure 85 and Figure 86, the chord shapes for the II7 and VI7 chords of the sequence are actually the same, but performed in different regions of the fretboard; note that both are barre--chords. The third chord shape (I7 degree chord) is much simpler; so simple that it does not require more than one finger of the left hand in order to be performed, as seen in Figure 87.

Figure 87: Fourth chord shape of the harmonic progression (degree I7)



Source: Own image.

Once the chord shape is established, the *IdiomaticGuitarist* uses the chord shape fingering information to calculate the VD and HD of each finger (movement). The movement is then passed to the appropriate (barre or non-barre) speed model to find the order of arrival of the fingers. The speed of each arrival will be predicted using its own model.

Table 31 shows the calculated transition movement between the barre-chords.

Table 31: Barre-chords transition movement.
The *Fret* represents the fingerboard location where the chord is meant to be performed, *nStr* is the number the strings the barre covers, and *HMM* is the hand motion. The other attributes are the vertical and horizontal displacements for the fingers. '*Mov.* ID' is just a label column and it is not used in the prediction.

Mov. ID	Fret	nStr	HMM	MVD	MHD	RVD	RHD	LVD	LHD
Mov(II7,VI7)	1	2	-5	0	-5	0	-5	0	-5
Mov(VI7,II7)	6	2	5	0	5	0	5	0	5

Unlike from the first three chords of the sequence, the last chord is not a barre-chord hence it must be described according to the input format of the non-barre chord models, shown in Table 32. Note that due to the simplicity of the chord shape, three fingers (M, R, and L) that were being used in the previous chord shape are no longer required. In this type of situation, it is impossible to calculate the fingers' vertical and horizontal displacements, so the instance attributes are signalised with a '-100' value, which tells the model they should not be considered.

Table 32: Attributes of the movement transition to a Non-barre chord.
I, M, R, and L represents the finger, Index, Middle, Ring and Little respectively.
VD = vertical displacement, HD = horizontal displacement.

Fret	IVD	IHD	MVD	MHD	RVD	RHD	LVD	LHD
1	1	0	-100	-100	-100	-100	-100	-100

Table 33 shows the predicted speed values (ms) for the transition of the chord shapes. The slowest predicted movement is the Mov (VI7,II7) taking 616 ms to be performed. As expected, the faster movement is Mov (II7,I7) as the index finger, one of the fastest fingers, must move to an adjacent position. The 'Barre (index)' column shows the predicted speeds for the barre position, where P1 = position in the first string, P2 = position in the second string.

Table 33: Speed model predictions.
Order is the predicted finger's arrival order. T1, T2 e T3 is the arrival time.

	Barre (index)	Order	T1	T2	T3
Mov(II7,VI7)	P1 – 229 ms P2 – 338 ms	RML	230	381	500
Mov(VI7,II7)	P1 – 385 ms P2 – 385 ms	RML	335	446	616
Mov(II7,I7)	-	IMRL	198	-	-

At this point, the speed values are used by the Idiomatic Guitarist as a limiter to the note duration. For example, in the Mov (VI7,II7) the little finger is predicted to take 616 ms to reach its position. If the RhythmPattern indicates that its respective duration is 1000 ms, then the note linked to the little finger will start with a 616 ms delay and will last for 1000 – 616 = 384 ms. If the duration is 500 ms, then the note is skipped (note deletion error).

Once the fingers' arrival order and timing have been predicted, the *IdiomaticGuitarist* move on to the force predictions. Once again, the prediction must use the appropriate models taking into consideration whether the chord shape uses a barre or not. Table 34 illustrates how the data must be prepared in order to find the force of the tip-pinch fingers (middle, ring, little) to barre-chord model.

Table 34: Format used to find the barre-chords force model.
The *Fret* represents the fingerboard location where the chord is meant to be performed, *nStr*ings is the number the strings the barre cover. The other attributes are the vertical and horizontal displacements for the fingers

Fret	nStrings	nNotes	MS	MF	RS	RF	LS	LF
6	2	2	0	0	4	8	5	9
1	2	2	0	0	4	3	5	4

Table 35 illustrates how the data must be prepared in order to find the force of the positions of the barre, executed by the index finger.

Table 35: Format used to find the barre positions force model (index finger). nStr is the number of strings the barre covers. nNotes is the number of notes the barre produces; pString is the string in which the force is related to. MS, MF, RS, RF, LS, LF are the string and fret coordinates for the middle, ring and little fingers respectively;

Fret	nStrings	nNotes	pString	MS	MF	RS	RF	LS	LF
1	2	2	1	0	0	4	3	5	4
1	2	2	2	0	0	4	3	5	4
6	2	2	1	0	0	4	8	5	9
6	2	2	2	0	0	4	8	5	9

The non-barre chord (I7) uses a slightly different and simplified format because the barre does not need to be specified. This is shown in Table 36.

Table 36: Format used to find Non-barre chords force models.
The columns represent the string and fret coordinates respectively for the middle, ring and little fingers;

Fret	IS	IF	MS	MF	RS	RF
1	3	1	0	0	0	0

With force values predicted, the *IdiomaticGuitarist* verifies if the force is enough to produce a clear note in the *IdiomaticGuitar* or a muffled/buzzed note should be produced instead. Table 37 and Table 38 show the fingering used to perform the chord shapes of the first (Figure 85) and second (Figure 86) chords of the harmonic sequence, where the 'Predicted' force is compared with the 'Required' force calculated to produce a clean note on the modelled guitar. The '%' column shows the force percentage ratio of the Predicted x Required, followed by the quality of the note that is simulated.

Table 37: The predicted force for the first chord shape of the harmonic sequence (II7 degree).
The '%' column shows the force percentage ratio of the Predicted x Required force which will determine the 'quality' of the predicted note.

String	Fret	Finger	Predicted	Required	%	Quality
1	6	Index (barre)	0.089	0.142	62	Muffled
2	6	Index (barre)	0.11	0.107	102	Clean
4	8	Ring	0.082	0.152	53	Muffled
5	9	Little	0.057	0.152	37.5	Muffled

Table 38: The predicted force for the second chord shape of the harmonic sequence (VI7 degree).

String	Fret	Finger	Predicted	Required	%	Quality
1	1	Index (barre)	0.145	0.303	47	Muffled
2	1	Index (barre)	0.263	0.230	114	Clean
4	3	Ring	0.063	0.163	38	Muffled
5	4	Little	0.046	0.144	31	Muffled

As noted, most of the predicted force to produce the notes is not enough to produce clean notes which would add a substantial amount of noise in this performance. To gain more control over the overall quality of the notes simulated, it is possible to adjust the predicted force by multiplying it by a 'confidence' constant factor. For example,

if the predicted force is multiplied by a factor of 0.6, then we would have three clean notes, two buzzed notes, and three muffled notes.

The precision errors are last to be calculated because this involves a completely different approach, as explained in Section 5.2.4 (p. 177).

The data used to train the models used in this example came from Subject 1, who had the smallest error rate of all three subjects. In order to recreate any error, the similarity threshold had to be brought down to zero. This means that the error search was too broad and did not produce errors that were 'convincing' from an empirical point of view.

In other simulations using the data from other subjects and even artificially created data, we observed that if the similarity threshold is bellow 0.5 then there is a tendency to produce errors that do not fit the reality. Of course, the most 'convincing' errors occur when the similarity threshold is equal to 1, but this would require a good number of examples in the database; the default value of 0.7 presents a good compromise.

It is important to note that the precision errors have their own timing and duration values that are extracted from the precision data. By convention, the default force applied in the precision errors is 50% of a clear note but this has little relevance at the moment because no sound can actually be produced yet, as explained in the next section.

OVERALL RESULTS AND FINAL CONSIDERATIONS

Whereas the results presented from the Machine Learning algorithms could be quantified using performance measures such as RMSE and RAE, our integrated solution can only be evaluated by means of a demonstration of how the Octopus Music API can be equipped with biomechanical-inspired models.

Naturally, the best approach to verify whether the models and algorithms actually achieve our goal would be to conduct perceptual

listening experiments. However, considering the technological limitations behind the guitar synthesisers that are currently available on the market, it is impossible to conduct a listening test at present. Nevertheless, this should not deter us from continuing research towards the future of musical performance by computers. To this end, we designed an alternative interim solution, which is to display the behaviour of the model visually. Hence the main reason we created the I*domaticGuitarGraphicalInterface* component of Octopus Music API.

As previously explained in 5.3.1.2, p. 211, the guitar GUI is capable of showing the note's pressure and precision errors based on a colour scheme. This is just enough to demonstrate that all the models work together. By way of further work, we can identify some aspects that need to be addressed to strengthen integration. The first of them refers to the lack of a force model for the little finger for non-barre chords. This could be solved by including the little finger in experiments involving recorded chord shapes. Another aspect requiring further work concerns the current assumption that the barre is performed only with the index finger. Although this is the case most of the time, there are some rare occasions when the barre might have to be performed with other fingers. A similar case is a use of the thumb as a fretting finger, which is considered wrong by the Classical School of Guitar but it has been used in more popular genres. These techniques can be modelled in the Octopus Music API, but the biomechanical-inspired models can not handle them at the moment.

Concerning the delays introduced by the speed models, we believe that a top-down approach could also produce good results. This means that, instead of applying delays at a note level, the delay could be spread throughout a musical phrase, a part, or even the entire music. However, this feature should be investigated together with the right hand, which is beyond this book.

More experiments to measure the speed of chord shapes that demand hand motion would also enrich our training data, consequently increasing the accuracy and prediction power of the models.

The use of meta-learners can produce a substantially more robust model but it also has disadvantages. In Section 5.2.2 (p. 169) we have explained the Stacking meta-learn scheme that is used to combine the prediction of a Decision Table, IBK and MP5 algorithms. In fact, the outcomes of these three inner-schemes are combined using a linear regression classifier. The problem with this approach is that when one of these schemes produces predictions that are significantly better than the others, it is highly weighted by the linear model and the influence of the others becomes irrelevant. The problem is aggravated when learning schemes perform an attribute selection and select attributes that are very good for the training dataset but are not good for the prediction of unforeseen examples. In these circumstances, it is better to sacrifice a low error rate to gain a more flexible model that provides better overall prediction.

The most efficient way to solve this problem is to invest time in data capture and preparation to make sure that a significant amount of representative instances are included in the training set, avoiding unforeseen instances that are radically different from anything the algorithm has drawn generalisations from. If this action does not produce the desired effect then the automatic selection of attributes of the meta-classifier (a linear regression in our case) may need to be disabled, which may increase the error rate slightly but also generate more robust models.

In the case of the precision errors, in which we have developed our own instance-based 'learning' algorithm designed to our particular context, there are other difficulties. Firstly, it requires manual adjustment from the 'user' to 'recreate' convincing errors. If the algorithm is not set up properly it can recreate errors that seem very unlikely to occur for the sake of producing an error.

For instance, if a large KNN number is set or a small similarity threshold, then errors from movements that are very different from the original may arise. Since we use absolute coordinates for errors, they may appear distant from the region where the chord is being

performed. Incompatibility of timing and duration is also likely. A possible solution is to use relative coordinates for the error positions but this would create other problems when the fingering is unknown or can not be determined. That said, the problem will only manifest itself if the similarity threshold or KNN is not set properly.

We look forward to the appearance of suitable Sound Generator Units on the market that is capable of rendering guitar performances with the auditory level required to simulate modelled errors, as proposed in this book. Our research has certainly provided good evidence that guitarists indeed do not all play guitar equally from a biomechanical perspective and this affects the performance. We sincerely hope that manufacturers of Sound Generator Units would soon take into account the merit of computational models, such as the ones proposed in this work, to model and replicate human behaviours during guitar performance.

SUMMARY

This Chapter was divided into two parts that approach two different but related problems in modelling music performance: a) the creation of a computational tool capable of describing and manipulating all the nuances of a musical performance from the performer viewpoint; and b) Machine Learning techniques that can extract patterns of force, speed and precision of the left-hand fingers when performing chords.

The Octopus Music API is a Java library designed to model music performance in the lowest possible level of abstraction. In its basic descriptive form, modelling a music performance can be a very time-consuming task because the slightest action in a performance must be explicitly declared. To overcome this issue, we have equipped *Music Interpreters Classes* (i.e. *Guitarist*) with the 'knowledge' (rules) to infer and adapt the performance actions to the context they are inserted. For instance, a *Guitarist* must adapt the music performance to fit the *Guitar* while a *Musician* has no such limitation.

It soon became obvious that a rule-based approach was not ideal to model highly complex scenarios, such as the biomechanical system. Hence, Machine Learning techniques were used to try to find patterns of speed, force and precision that could lead to errors in guitar performance. The resulting models were integrated into the Octopus Music API by extending the *Guitar* and *Guitarist* classes with its idiomatic counterparts: *IdiomaticGuitar* and *IdiomaticGuitarist*.

At this stage, we can demonstrate that our modelling approach works. However, integration with a Sound Generation Unit, capable of rendering the predicted imperfection in performance into sound, is the next step towards further research into the role of performance in computer-generated music performance research.

Chapter 6

Conclusion

> The reasonable man adapts himself to the world; the unreasonable one persists in trying to adapt the world to himself. Therefore, all progress depends on the unreasonable man. (George Bernard Shaw).

Some researchers believe that to simulate a truly expressive and humanised music performance we must first understand the way we think, perceive, and feel the music. The composer, the interpreter, and the listener have all been under the scrutiny of scientific investigation that led to the creation of several theories in music cognition (Juslin and Sloboda, 2001; Palmer, 1997; Shepard, 2002; Sloboda, 2000; Todd, 1989b). We have called this approach behavioural-based because they focus on proving cognitional theories that ultimately can be used to create computer-generated music performance. This approach was discussed in Section 2.3 (p. 30).

Another approach used to program machines to perform music is to mimic human behaviour without really paying attention to the underlying processes that trigger certain performance actions. This approach we have called simulation-based because it focuses on the

output (simulation) rather than the behaviour, as opposed to the behaviour-based models. It is in this category that the current computational techniques thrive, including the state of the art AI techniques, as seen in Section 2.2 (p. 27);

As Sundberg (2000) observed, psychological studies of music performance have provided a wealth of information on musical expression but they have largely ignored the physical manipulation of the instrument by the performer. The interaction between humans and artefacts has been studied in disciplines such as ergonomics, biomechanics, and human factor sciences even though these studies rarely focus on music performance modelling. In reality, just a few studies consider the influence of the body in models for musical performance. We have discussed two of them in Section 2.3.5 (p. 35).

It is at the physical level that accidental errors happen, known as slips. It is a well-known fact in the field of biomechanics that motions can be made more rapidly in certain ways and directions because of the nature of the human physical structures (Rosenbaum, 1996). These physical structures can limit the movement speed which would eventually induce errors. The biomechanical, physiological, and anatomical properties of the human body during a guitar performance were presented in Section 3.2 (p. 64).

We have seen that muscle strength, speed, and endurance can indeed affect music performance. For instance, we have shown that the index finger needs at least 20 ms to generate enough power to produce a clean note in a guitar (Section 3.2.3.3, p. 79). Curiously, this is the same amount of time that Pisoni (1977) reported for listeners to be able to distinguish temporal differences between two successive acoustic events.

The forces required to deflect a string in a real guitar were both calculated and measured; the average calculated force to produce a clean note was found to be 223 grams and the average measured force was 423 grams. If not enough force is applied to stop the string, a muffled or a buzzed note is likely to be produced instead. De facto, we have found that a buzzed-note requires on average 75% of the

force necessary to produce a clean note. The mechanics of the guitar, including the string deflection calculations and measurements, were discussed in Section 3.1 (p. 47).

Muffled and buzzed notes are especially relevant to this book because they are the direct result of the finger's inappropriate use of force. Unfortunately, these two particular 'noises' have not yet been supported by modern synthesis techniques (not even those based on physical modelling techniques), although it is acknowledged that noise can be an important part of instrument tone, as explained in Section 2.5 (p. 42).

Even if currently available synthesizers were able to support noise in musical performances, there would still be the problem of controlling it; for instance, when and how they should occur. By 'noise' here we mean the result of those unintentional actions originating from the motor and biomechanical functions that we have discussed throughout this book. When we started this research, nothing could be found in the literature addressing this issue. Much investigation is required to ascertain when a note should have its sound quality intact or some form of 'noise modulation' should be produced instead. In an attempt to understand that, we have designed a set of experiments to measure not only the force produced in a multi--finger task (playing a guitar chord) but also its speed and precision.

Since an integrated measuring device capable of recording the force, speed and precision of the finger in a guitar performance does not exist we had to design and build our own device: FoGu. The full description of the experiments is reported in 0 (p. 47).

The speed results have shown that certain chords can be performed twice as fast as others, with the average speed required for a chord to be performed around 350 ms. As expected, chord shapes that better suit the hand's anatomy, such as A and E chords, presented a smaller speed variation between the subjects, respectively at 36 and 24 ms. This evidence contributes to the belief that biomechanical constraints can indeed delay some actions in music performance. All the speed results are found at Section 4.2.4 (p. 96)

The force results have shown that the average force distribution among the fingers is slightly different from what is found in the literature, where the middle finger is usually the main force producer. In our experiment, the index finger was the main contributor with 32% of force produced, followed the middle, ring and little fingers with 30, 21 and 17% respectively. The average force the guitarists believed was necessary to produce a clean note was around 147 grams/f. The full analysis of the force measurements can be seen in Section 4.3.4.1 (p. 116).

The posture and motion analysis revealed surprising results. From the three articulations measured (wrist, elbow and shoulder), the elbow was the one which presented the highest level of motion. We initially believed that the wrist would be more important. These results must be handled with care because the equipment used was limited to measuring just a few degrees of freedom of articulation, especially the wrist movements. A full discussion of the posture results is presented in Section 4.3.4.2 (p. 125)

Although the results of the experiments have disclosed interesting evidence to support the notion that biomechanical constraints indeed interfere with music performance, we have opted to not translate these findings straight to production rules that could be used to simulate music performance. Instead, we adopted a machine learning approach. We envisage that in the future systems should be able to learn these rules and for this reason, we used machine learning (ML).

In order to choose suitable ML algorithms for the tasks at hand, we adopted a toolbox approach: several algorithms were tried and those that performed better were selected. An open-source framework known as WEKA (Witten and Frank, 2002) was essential for this job.

Before the data could be presented to the ML algorithms, it had to be prepared, this involved the removal of outliers, attribute selection, and even the creation of attributes that highlight relationships. The data preparation for barre and non-barre chords was done separately due to the biomechanical difference in the type of handgrip required to perform them.

To predict the speed we used a combination of numeric prediction learning algorithms: instance-based IBK1, a Decision Table, and Mode tree (MP5). The outputs of these three algorithms were combined using the Stacking meta-learner that used a Linear Regression as the classifier.

The average learning error rate for non-barre chords was 10.41% RAE with an average confidence of over 99%. For the barre-chords, the RAE was on average 9.86% with confidence also over 99%.

Because speed refers to a movement (dynamic) instead of a static posture, we model the 'arrivals' instead of the fingers; all three 'arrival' models were modelled independently from each other. The arrival time is only relevant if the finger's arrival order can also be predicted. This was done using a combination of classification learning algorithms: Random Forest, Random Committee/Random Tree, and instance-based IBK. The predictions were combined using Bagging (voting) meta-learning, so the algorithms were selected based on their results rather than on the way they theoretically suit the data.

Using the combination of algorithms mentioned above we have achieved positive classifications of 97.7% for non-barre chords and 99% for barre chords. The full explanation of the modelling of speed data can be found at Section 5.2.4 (p. 177).

The models for force prediction were created using a similar procedure used to model the speed, but instead of modelling the 'arrival' we modelled the force of the fingers because there was no movement involved, in other words, it was a static effort in which the fingering was known beforehand. Each finger was modelled independently.

The learning algorithms were the same used for speed modelling, with one exception: we replaced the model tree with another tree of numerical prediction known as a regression tree (REPTree). They work similarly, but the REPTree produced better results with the force data.

The learning rates for the force data were not as good as for speed, averaging 20% RAE with average confidence over 97% for

non-barre chords; barre-chords had an average RAE of 17.8% with confidence also over 97%. We believe that learning performance was harmed by noisy data caused by technical problems with the force sensors, as reported in Section 5.2.5 (p. 189).

The precision data required different treatment due to the scarce number of examples. Instead of using a standard learning technique of WEKA, we decided to implement a custom instance-based (k-Nearest-Neighbour) algorithm. The advantages and disadvantages of this approach are discussed in Section 5.2.6 (p. 200).

In order to try the models in a musical context, the speed and force models were integrated into framework designed to simulate musical performance. This framework, named Octopus Music API, has also served as the basis for the implementation algorithm that recreates precision errors.

The main idea behind a framework like the Octopus Music API is to facilitate the modelling of music performance by transferring some of the knowledge involved in adjusting a piece to music to a particular instrument to the system. This was done by creating three categories of classes to represent the performance elements: a) Musical Data Structure; b) Musical Data Interpreters; and c) Musical Instrument Classes.

Musical Data Interpreter classes, such as the *Guitarist*, know how to read the Musical Data Structures (i.e. *Music, Harmony, Chord etc)* and adapt the musical information to the limitations of the Instrument (i.e. Guitar), as shown in Section 5.1, p. 135. The force, speed and precision modelling have extended this adjustment to also consider some of the performer's physical limitations. The integration of the Octopus Music API with the biomechanical-inspired model was discussed in Section 5.3 (p. 208).

CONTRIBUTIONS TO KNOWLEDGE

The contributions to the knowledge of this book are presented below in two sections. Firstly we indicate how the book answered the three

overarching motivation questions posed in Chapter 1. Then, further contributions to knowledge are indicated in the context of the five research objectives, also introduced in Chapter 1

Answers to the Motivation Questions

Do unintentional actions originating from the motor and biomechanical functions during musical performances contribute to the 'human feel' found in the performance?

As demonstrated in Sections 4.2.4 (p. 96) and 4.3.4 (p. 116) the answer to this question is yes. We proved that the unintentional actions originating from the motor and biomechanical function do impact the quality of music performance. We understand unintentional actions as slips, a category of error in which the correct action was planned (cognitive side) but, for some reason, it was not delivered. By modelling the motor and biomechanical system we can, at least in part, recreate the human aspect of a musical performance by reintroducing the errors that are lacking in computer-generated musical performances.

Would it be possible to determine and quantify what such unintentional actions are?

Again, the answer to this question is yes. In 0 (p. 47) we presented a methodology for acquiring and analysing the data for such unintentional actions in guitar performance. We focused on measuring the speed, force, and precision of the guitarist left-hand when performing the chords.

We have used three experienced guitarists in our experiments. The tasks in the experiment involved the performance of simple and familiar chord shapes. To measure the speed and precision data we used a guitar-like MIDI controller. The force was measured on a custom-build device shaped in the form of a guitar.

The results obtained confirmed that the level of effort required to perform certain actions in guitar performance can induce errors. Moreover, it showed that subjects have a distinct way to perform those actions, making them more prone to certain types of errors than others.

Would it be possible to model and embed such information in computer systems for music performance?

Yes, it is possible to model and use this information in computer systems to simulate music performance, as demonstrated in 0 (p. 131). The modelling of force and speed was done using a combination of ML algorithms that achieved a combined average of Relative Absolute Error (RAE) for predictions under 15%, with confidence over 98%. The algorithms, the data preparation procedure, the training strategy, and the results are described in Section 0 (p. 164).

We have also presented the Octopus Music API, which is a library designed to model musical performances (Section 5.1, p. 135). The biomechanical-inspired models were incorporated into the Octopus Music API so performance errors can be predicted and automatically inserted into a simulated guitar performance without having to be manually specified. More details of this integration are found in Section 5.3 (p. 208).

Approach to the book's overreaching goals

Understand gaining of the guitar mechanics, ergonomics, and playability.

In Chapter 3 (Section 3.1, p. 47) we presented a study supported by extensive reference to relevant literature on the mechanics, ergonomics and playability of the guitar. The focus of the study was on the physical actions that must be performed on the guitar to produce sounds, rather than the acoustical properties of the sound. One

particular type of sound resulting from the performer's motor and biomechanical limitations was covered in-depth, the noises from muffled and buzzed noted (Section 3.1.6, p. 56).

The understanding gained of how the human body conforms to physical actions in a musical performance.

Another aspect covered by Chapter 3 (Section 3.2, p. 64) is how the human body conforms to physical actions in guitar performance from the biomechanical and physiological point of view. It is the first time that a study of this kind has focused on guitar performance.

To understand the stress that the body is subjected to in guitar performance, we first established the movements and postures required by the classical guitar technique (Section 3.2.2, p. 66). Then the muscles and articulations involved in those physical actions were studied to determine the power, speed and precision that they are capable of delivering (Section 3.2.3, p. 74).

Development of a methodology to formalise quantifiable data about physical performing actions found in guitar performance.

In 0 (p. 47) we proposed a methodology to record and process the force, speed, and precision of the guitarist's left-hand when performing chords shapes. It is the first time that a multi-finger task related to guitar performance has been measured.

The speed and precision were already partially measured by Heijink and Meulenbroek (2002) but they were limited to single-finger tasks (playing a scale). Results and comparison with this study were reported in Section 4.2.4 (p. 96). Force measurements in guitar performance could not be found in the literature. Perhaps because a commercial measuring tool suitable for the task does not exist. In Section 4.3 (p.106) we described how we captured this data and what they revealed.

An approach to model the biomechanical data.

In 0 (p. 131) we demonstrated an approach to model the speed, force, and precision data using Machine Learning (ML) techniques. The obvious advantage of the ML over the rule-based approach is that the ML algorithms automatically learn the guitarist's physical and performing differences, allowing the system to model a particular individual rather than search for generalised rules that cover most of the cases.

As previously said, a vast amount of effort in data mining is dedicated to data preparation. By presenting a successful way of doing it, we have significantly contributed to any research that may follow.

Demonstration of how the proposed modelling approach can be embedded in a computer system for music performance

In the first section 0 (Section 5.1, p. 135) we proposed an approach to developing systems for music performance, which give the users the power to implement such systems themselves; to implement bespoke systems addressing their specific needs. As result, we designed Octopus Music API, which is a Java library that introduces a unique way to model musical performance by specifying not only the musical but also the performer and instrument aspects involved in the performance.

By modelling the three main elements of a music performance – the performer, the instrument, and the music - the Octopus Library is capable of automatically adjusting the performance actions according to the limitations of each of those elements, so the user does not have to do so. This approach is better because – for example – different styles of music may require different performance strategies; different musical instruments require different performance information and indeed, different performers perform music differently.

The mechanics of the guitar described in Section 3.1 (p. 47) and the biomechanical aspects of the performer, described in Section 3.2 (p. 64)

and measured in 0, p. 47, were incorporated into the Octopus through the use of Machine Learning techniques described in Section 0 (p. 164). A full example of this integration was presented in Section 5.3 (p. 208)

Final thoughts on the future of the field

We have already mentioned several times throughout this book that a full verification of the impact of the errors in a computer-generated musical performance is only possible when the acoustical properties of the errors, in the form of noises, can also be modelled. So, the first recommendation for future works is the integration of *Idiomatic* classes of the Octopus Music API with a Sound Generation Units that can handle the sound synthesis in the level of detail modelled by the Octopus, such as the work of Bader(2009), Erkut *et al. (2000),* Laurson *et al. (2001),* Laurson *et al.* (2005), Poepel, (2004a), and Valimaki *et al.*(1996) among others.

Another step forward would be the integration of the biomechanical-inspired models with the cognitive-inspired models of musical performance, such as those proposed by Thompson (2006), Sloboda (2000), or Todd (1989b).

An important aspect that has been left out of our analysis is right-hand. The right-hand has a significant role in controlling performance timing and consequently the delays. However, the synchronism of the hands must be investigated for precise use of motor and biomechanical delays. The right-hand is also responsible for sound 'intensity' and consequently the accentuation of the notes. Naturally, this is also subjective to errors that were not in the scope of this research.

An issue that has not been appropriately exploited by our experiments is the correlation between posture and force. This was due to the lack of a synchronisation mechanism between the force and posture measuring devices. Since this correlation could not be finely established, we opted to not use the posture data at this stage.

Regarding speed and precision modelling, we would like to implement them in the form of a continuous learning process. Hence, instead of passing pre-generated models as arguments to the *IdiomaticGuitarist*, the algorithms would be embedded in it and would be generated at the moment of instantiation or whenever requested. This feature would make possible the use of an inexpensive MIDI guitar to continuously feed the models. Fatigue, learning rate, musical style, and other factors could then have their relevance accessed and incorporated into the models.

An update of the force model may not be feasible because of the specialised device used to record the data. Therefore, more experiments could be beneficial, although, in data mining, more data is not necessarily good for learning. Even though the amount of data at our disposal was adequate to validate the proposed methodology, in retrospect we would like to have chosen more non-barre chords that make use of the little finger. The hand motion effect is another aspect that could have been investigated further with more examples. Perhaps, the way forward consists of using data captured from real guitar performances.

Finally, the development of the Octopus Music API is a continuous process, since it has been freely available and used by the community since 2007. We are keen to welcome new developers by turning the Octopus into an open-source project. One of the most interesting suggestions made by the community of users includes the creation of a graphical programming interface which would make the Octopus more user-friendly to non-programmers users and fit to live coding performances.

Bibliography

Aha, D. W., Kibler, D. and Albert, M. K. 1991. Instance-Based Learning Algorithms. **Machine learning**, 6, (1) 37--66.

Animazoo. 2009. Gypsy-6. Available online at: http://www.animazoo.com/Gypsy6.aspx. Accessed: September 2009

Bader, R. 2009. Real Time Guitar Radiation Sound Synthesis of Forced String and Body Eigenfrequency Vibrations Using Microphone Array Techniques. **The Journal of the Acoustical Society of America**, 125, (4) 2515.

Baily, J. 1985. **Music Structure and Human Movement**. Academic Press, London , UK. 237-58 pp.

Bean, K. L. 1939. Reading Music Instead of Spelling It. **Journal of Musicology**, 1, (1) 1-5.

Bejjani, F. J. and Halpern, N. 1989. Postural Kinematics of Trumpet Playing. **Journal of Biomechanics**, 22, (5) 439--446.

Bellman, R. 1961. **Adaptive Control Processes: A Guided Tour**. Princeton University Press, Princeton, New Jersey, U.S.A.

Bennett, A. and Dawe, K. 2001. **Guitar Cultures**. Berg, Oxford, UK.

Beran, J. and Mazzola, G. 1999. Analyzing Musical Structure and Performance - a Statistical Approach. **Statistical Sciefnce**, 14, (1) 47--79.

Bilitski, J. 2005. **A Machine Learning Approach for Automatic Performance of a Trumpet**. Selvaraj, H., Verma, B. and DeCarvalho, A. (Eds). 6th International Conference on Computational Intelligence and Multimedia Applications. Las Vegas, NV. Aug 16-18. pp 80-85.

Booch, G., Maksimchuk, R., Engle, M., Young, B., Conallen, J. and Houston, K. 2007. **Object-Oriented Analysis and Design with Applications**. Addison-Wesley Professional

Breiman, L. 2001. Random Forests. **Machine learning**, 45, (1) 5--32.

Burns, A.-M. and Wanderley, M. 2006. Visual Methods for the Retrieval of Guitarist Fingering. New interfaces for musical expression. IRCAM - Centre Pompidou, Paris, France.

Cabena, P., Stadler, R. and Zanasi, A. 1998. **Discovering Data Mining: From Concept to Implementation**. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Card, S. K., Moran, T. P. and Newell, A. 1983. **The Psychology of Human-Computer Interaction**. L. Erlbaum Associates, Hillsdale, N.J.

Carlevaro, A. 1984. **School of Guitar: Exposition of Instrumental Theory**. Boosey & Hawkes.

Chao, E. Y. 1989. **Biomechanics of the Hand: A Basic Research Study**. World Scientific Publishing Company.

Chapman, R. 1994. **The Complete Guitarist**. Dorling Kindersley.

Clarke, E. F. 1988. Generative Principles in Music Performance. **Generative Processes in Music: The Psychology of Performance, Improvisation, and Composition**. Clarendon Press/ Oxford University Press, New York, NY. pp 1-26.

Clarke, E. F. 1993. Generativity, Mimesis and the Human Body in Music Performance. **Contemporary Music Review**, 9, (1) 207--219.

Costalonga, L. and Viccari, R. M. 2004. **Multiagent System for Guitar Rhythm Simulation**. International Conference on Computing, Communications and Control Technologies. Austin, Texas, USA.

Costalonga, L. L., Miletto, E. M., Flores, L. V. and Vicari, R. M. 2005. Bibliotecas Java Aplicadas a Computação Musical. Simposio Brasileiro de Computacao Musical. Belo Horizonte – MG, Brazil.

Costalonga, L. L. and Miranda, E. R. 2006. Idiomatic Guitar Synthesis. **Journal of the Acoustical Society of America**, 119, (5) 3441.

Costalonga, L. L., Vicari, R. M. and Miletto, E. M. 2008. Agent-Based Guitar Performance Simulation. **Journal of the Brazilian Computer Society**, 14, (3) 19--29.

Cuzzucoli, G. and Lombardo, V. 1999. A Physical Model of the Classical Guitar, Including the Player's Touch. **Computer Music Journal**, 23, (2) 52--69.

Dahl, S. 2006. Movements and Analysis of Drumming. In: Altenmuller, Eckhart, Wiesendanger, Mario, Kesselring and Jurg (Eds). **Music, Motor Control and the Brain**. Oxford University Press, New York, NY, US. pp 125--138.

Danion, F. and Galléa, C. 2004. The Relation between Force Magnitude, Force Steadiness, and Muscle Co-Contraction in the Thumb During Precision Grip. **Neuroscience Letters**, 368, (2) 176--180.

Dannenberg, R. B. 1993. A Brief Sruvey of Music Representation Issues, Techniques, and Systems. **Computer Music Journal**, 17, (3) 20--30.

Dannenberg, R. B. 1997. The Implementation of Nyquist, a Sound Synthesis Language. **Computer Music Journal**, 21, (3) 71--82.

Das, M., Howard, D. M. and Smith, S. L. 2001. The Kinematic Analysis of Motion Curves through Midi Data Analysis. **Organised Sound**, 4, (3) 137--145.

De Poli, G. 2004. Methodologies for Expressiveness Modelling of and for Music Performance. **Journal of New Music Research**, 33, (3) 189.

Dell, G. S. 1986. A Spreading-Activation Model of Retrieval in Sentence Production. **Psychological Review**, 93, (3) 283--321.

Denyer, R. 1992. **The Guitar Handbook**. (2 ed) Pan Bks., London.

Desain, P. and Honing, H. 1992. **Music, Mind and Machine: Studies in Computer Music, Music Cognition and Artificial Intelligence**. Thesis, Amsterdam.

Desain, P. and Honing, H. 1994. Does Expressive Timing in Music Performance Scale Proportionally with Tempo? **Psychological Research**, 56, (4) 285--292.

Desain, P., Honing, H. and Heijink, H. 1997. **Robust Score-Performance Matching: Taking Advantage of Structural Information**. International Computer Music Conference. Thessalonki, Greece. pp 337--340.

Dietterich, T. G. 2000. Ensemble Methods in Machine Learning. **Lecture Notes in Computer Science**. Springer Berlin Berlin, Germany. pp 1--15.

Dixon, S., Goebl, W. and Widmer, G. 2002. **The Performance Worm: Real Time Visualisation of Expression Based on Langner's Tempo-Loudness Animation**. International Computer Music Conference Goteborg, Sweeden.

Dixon, S., Goebl, W. and Widmer, G. 2005. The "Air Worm": An Interface for Real-Time Manipulation of Expressive Music Performance. **Proceedings of the International Computer Music Conference (ICMC'2005)**,

Dodig-Crnkovic, G. 2002. Scientific Methods in Computer Science. Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden. Skovde, Sweden. Available online at: http://www.mrtc.mdh.se/index.php?choice=publications&id=0446.

Dogantan-Dack, M. 2006. The Body Behind Music: Precedents and Prospects. **Psychology of Music**, 34, (4) 449--464.

Doyle, J. R., Botte, M. J. and Krames, C. 2003. **Surgical Anatomy of the Hand and Upper Extremity**. Lippincott Williams & Wilkins Philadelphia.

Drake, C. and Palmer, C. 1993. Accent Structures in Music Performance. **Music Perception**, 10, (3) 343--378.

Edwards, B. 1983. **Fretboard Logic: The Reasoning Behind the Guitar's Unique Tuning** Edwards Music Pub, Temple Terrace, Florida - USA.

Embry, D. E. 1986. Sherpa: A Systematic Human Error Reduction and Prediction Approach. International Topical Meeting on Advances in Human Factors in Nuclear Power Systems. Knoxville, TN.

Ericsson, K. A. 1993. The Role of Deliberate Practice in the Acquisition of Expert Performance. **Psychological Review**, 100, (3) 363--406.

Erkut, C., Valimaki, V., Karjalainen, M. and Laurson, M. 2000. Extraction of Physical and Expressive Parameters for Model-Based Sound Synthesis of the Classical Guitar. 108th Audio Engineering Society Convention. New York, NY.

Evans, T. and Evans, M. A. j. a. 1977. **Guitars: Music, History, Construction and Players from the Renaissance to Rock / Tom and Mary Anne Evans**. Paddington Press: distributed by Grosset & Dunlap, New York :.

Farga, F. 1969. **Violins & Violinists**. Barrie & Jenkins.

Faulkner, J. A., Claflin, D. R. and McCully, K. K. 1986. Power Output of Fast and Slow Fibers from Human Skeletal Muscles. In: N.L. Jones,

N.M., and A.J. Comas (Ed). **Human Muscle Power**. Human Kinetics Pub Champaign, IL. pp 81--91.

Faulkner, J. A., Jones, D. A., Round, J. M. and Edwards, R. H. T. 1980. **Dynamics of Energetic Processes in Human Muscle**. International Symposium on Exercise Bioenergetics and Gas Exchange. Milan, Italy. Elsevier-North-Holland Biomedical Press. pp 81.

Fernández, L. and Rodemann, N. R. 2005. **Flamenco Music Theory: Rhythm, Harmony, Melody, Form**. Acordes Concert Sl.

Fitts, P. M. 1954. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. **Journal of Experimental Psychology**, 47, (3) 381--391.

Fix, E. 1951. Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. Reprinted in Silverman, Bw and Mc Jones (1989), E. Fix and Jl Hodges (1951):" an Important Contribution to Nonparametric Discriminant Analysis and Density Estimation". **International Statistical Review**, 57, (3) 233-–247.

Fraisse, P. 1982. Rhythm and Tempo. In: Deutsch, D. (Ed). **The Psychology of Music**. Lawrence Erlbaum Associates. pp 149--180.

Freitas, A. A. 2002. **Data Mining and Knowledge Discovery with Evolutionary Algorithms**. Springer Verlag.

Freivalds, A. 2004. **Biomechanics of the Upper Limbs: Mechanics, Modeling, and Musculoskeletal Injuries**. CRC Press.

Friberg, A. 1995. **A Quantitative Rule System for Musical Performance. Doctoral Thesis**. Royal Institute of Technology, Stockholm.

Friberg, A., Bresin, R. and Sundberg, J. 2006. Overview of the Kth Rule System for Musical Performance. **Advances in Cognitive Psychology**, 2, (2-3) 145--161.

Gabrielsson, A. 1999. The Performance of Music. In: Deutsch, D. (Ed). **The Psychology of Music**. Academic Press, San Diego, CA, US. pp 501--602.

Gareth, L. and Curtis, A. 1985. Programming Languages for Computer Music Synthesis, Performance, and Composition. **ACM Comput. Surv.**, 17, (2) 235--265.

Garrett, M. F. 1980. Levels of Processing in Sentence Production. In: Butter-worth (Ed). **Language Production**. Academic Press, London, UK. pp 177--220.

Gilbert, J., Ponthus, S. and Petiot, J. F. 1998. Artificial Buzzing Lips and Brass Instruments: Experimental Results. **The Journal of the Acoustical Society of America**, 104, (3) 1627.

Gilden, D. L. 2001. Cognitive Emissions of 1/F Noise. **Psychological Review**, 108, (1) 33-56.

Gilden, D. L., Thornton, T. and Mallon, M. W. 1995. 1/F Noise in Human Cognition. **Science**, 267, (5205) 1837.

Grandjean, E. 1988. **Fitting the Task to the Man**. Taylor & Francis, New York.

Hago. 2009. Guitar String Deflection Calculations. Available online at: http://www.hago.org.uk/faqs/formulae-2.php. Accessed: September 2009.

Haslinger, B., Erhard, P., Altenmüller, E., Hennenlotter, A., Schwaiger, M., Gräfin von Einsiedel, H., Rummeny, E., Conrad, B. and Ceballos-Baumann, A. O. 2004. Reduced Recruitment of Motor Association Areas During Bimanual Coordination in Concert Pianists. **Human Brain Mapping**, 22, (3) 206--215.

Heijink, H. and Meulenbroek, R. G. J. 2002. On the Complexity of Classical Guitar Playing: Functional Adaptations to Task Constraints. **Journal of Motor Behavior**, 34, (4) 339--351.

Henderson, M. T. 1936. Rhythmic Organization in Artistic Piano Performance. In: Seashore, C.E. (Ed). **Objective Analysis of Musical Performance**. Univ. of Iowa Studies in the Psychology of Music, Iowa City: University of Iowa. pp 281--305.

Heuer, H. 1991. Invariant Relative Timing in Motor-Program Theory. **Advances in psychology**, 81, 37--68.

Hill, A. V. 1938. The Heat of Shortening and the Dynamic Constants of Muscle. **Proceedings of the Royal Society of London. Series B**, . Biological Sciences, London. pp 136--195.

Hogan, N. 1985. The Mechanics of Multi-Joint Posture and Movement Control. **Biological Cybernetics**, 52, (5) 315--331.

Honing, H. 2001. From Time to Time: The Representation of Timing and Tempo. **Computer Music Journal**, 25, (3) 50--61.

Honing, H. 2003. The Final Ritard: On Music, Motion, and Kinematic Models. **Computer Music Journal**, 27, (3) 66--72.

Honing, H. 2006a. Computational Modeling of Music Cognition: A Case Study on Model Selection. **Music Perception**, 23, (5) 365--376.

Honing, H. 2006b. Evidence for Tempo-Specific Timing in Music Using a Web-Based Experimental Setup. **Journal of Experimental Psychology: Human Perception and Performance**, 32, (3) 780--786.

Iberall, T., Preti, M. J. and Zemke, R. 1989. Task Influence on Timing and Grasp Patterns in Human Prehension. **Society of Neuroscience Abstracts**, 15, (1) 397.

Instron. 2009. Instron Universal Testing Machines. Available online at: http://www.instron.co.uk/wa/products/universal_material/5560.aspx. Accessed: September 2009.

Jacobs, J. P. 2001. Refinements to the Ergonomic Model for Keyboard Fingering of Parncutt, Sloboda, Clarke, Raekallio, and Desain. **Music Perception**, 18, (4) 505--511.

Jebsen, R. H., Taylor, N., Trieschmann, R. B., Trotter, M. J. and Howard, L. A. 1969. An Objective and Standardized Test of Hand Function. **Archives of Physical Medicine and Rehabilitation**, 50, (6) 311.

Johansson, R. S. 1996. Sensory Control of Dexterous Manipulation in Humans. **Hand and Brain: The Neurophysiology and Psychology of Hand Movements**. pp 381–414.

Johns, M. V. 1961. An Empirical Bayes Approach to Non-Parametric Two-Way Classification. In: Solomon, H. (Ed). **Studies in Item Analysis and Prediction**. Stanford University Press. pp 221--232.

Johnson, E. W. and Olsen, K. J. 1960. Clinical Value of Motor Nerve Conduction Velocity Determination. **Journal of the American Medical Association**, 172, (18) 2030--2035.

Juslin, P. N. 2003. Five Facets of Musical Expression: A Psychologist's Perspective on Music Performance. **Psychology of Music**, 31, (3) 273--302.

Juslin, P. N., Friberg, A. and Bresin, R. 2002. Toward a Computational Model of Expression in Music Performance: The Germ Model. **Musicae Scientiae**, 6, (1) 63--122.

Juslin, P. N. and Sloboda, J. A. 2001. **Music and Emotion Theory and Research**. Oxford University Press, Oxford.

Karjalainen, M., Välimäki, V. and Janosy, Z. 1993. Towards High--Quality Sound Synthesis of the Guitar and String Instruments. International Computer Music Conference. Tokyo, Japan. 56-63.

Kendall, R. A. and Carterette, E. C. 1990. The Communication of Musical Expression. **Music Perception**, 8, (2) 129-164.

Kohavi, R. 1995. **The Power of Decision Tables**. Proceedings of the European Conference on Machine Learning. Springer Verlag. pp 174-189.

Kong, Y. K. and Freivalds, A. 2003. Evaluation of Meat-Hook Handle Shapes. **International Journal of Industrial Ergonomics**, 32, (1) 13-23.

Kumar, S. 2004. **Muscle Strength**. CRC Press, Danvers, MA, USA.

Kuncheva, L. I. 2004. **Combining Pattern Classifiers: Methods and Algorithms**. Wiley-Interscience, Hoboken, NJ, USA.

Laurson, M. 2000. Real-Time Implementation and Control of a Classical Guitar Synthesizer in Supercollider. **Proceedings of the 2000 International Computer Music Conference**, 74-77.

Laurson, M., Erkut, C., Valimaki, V. and Kuuskankare, M. 2001. Methods for Modeling Realistic Playing in Acoustic Guitar Synthesis. **Computer Music Journal**, 25, (3) 38-49.

Laurson, M., Norilo, V. and Kuuskankare, M. 2005. Pwglsynth: A Visual Synthesis Language for Virtual Instrument Design and Control. **Computer Music Journal**, 29, (3) 29-41.

LeVan, J. 2005. **Classic Guitar Care and Setup**. Mel Bay Publications.

Loy, G. and Abbott, C. 1985. Programming-Languages for Computer Music Synthesis, Performance, and Composition. **Computing Surveys**, 17, (2) 235-265.

Luger, G. F. 2002. **Artificial Intelligence: Structures and Strategies for Complex Problem Solving**. (5th ed) Addison Wesley Publishing Company, Harlow, UK.

Lysloff, R. T. A. and Matson, J. 1985. A New Approach to the Classification of Sound-Producing Instruments. **Ethnomusicology**, 29, (2) 213-236.

MacKenzie, C. L. and Van Eerd, D. L. 1990. Rhythmic Precision in the Performance of Piano Scales: Motor Psychophysics and Motor Programming. **Attention and Performance Xiii**. Lawrence Erlbaum Associates, Hillsdale, NJ, USA. pp 375-408.

Madsen, S. T. and Widmer, G. 2006. Exploring Pianist Performance Styles with Evolutionary String Matching. **International Journal on Artificial Intelligence Tools**, 15, (4) 495-513.

Marshall, M. M., Mozrall, J. R. and Shealy, J. E. 1999. The Effects of Complex Wrist and Forearm Posture on Wrist Range of Motion. *Human factors*, 41, (2) 205.

Mathews, M. V. and Moore, F. R. 1970. Groove—a Program to Compose, Store, and Edit Functions of Time. **Communications of the ACM**, 13, (12) 715-721.

Mathiowetz, V., Kashman, N., Volland, G., Weber, K., Dowe, M. and Rogers, S. 1985. Grip and Pinch Strength: Normative Data for Adults. **Arch Phys Med Rehabil**, 66, (2) 69-74.

Maurer, J. J., Singer, M. A. and Schieber, M. H. 1995. Fiber Type Composition of Morphologic Regions in the Macaque Multitendoned Finger Muscles. **Acta Anatomica**, 154, (3) 216-223.

Mazzola, G. 2002. Performance and Interpretation. **Journal of New Music Research**, 31, (3) 221-232.

McCartney, J. 2002. Rethinking the Computer Music Language: Supercollider. **Computer Music Journal**, 26, (4) 61-68.

Meister, D. 1989. **Conceptual Aspects of Human Factors**. Johns Hopkins University Press Baltimore.

Meyer, J. 1983. **Quality Aspect of the Guitar Tone**. Committee for the Acoustics of Music Conference. Royal Swedish Academy of Music. pp 51-75.

Mitchell, T., Buchanan, B., DeJong, G., Dietterich, T., Rosenbloom, P. and Waibel, A. 1990. Machine Learning. **Annual Reviews in Computer Science**, 4, (1) 417--433.

Mizuno, M. 1994. P-31-Nmr Spectroscopy, Rsemg, and Histochemical Fiber Types of Human Wrist Flexor Muscles. **Journal of Applied Physiology**, 76, (2) 531-538.

Mizuno, T. and Takashima, S. 2001. Development of an Automatic Playing Robot of Bamboo Flute. Experimental Analysis of Fingering System and Artificial Mouth. **Nippon Kikai Gakkai Robotikusu, Mekatoronikusu Koenkai Koen Ronbunshu**, 2001, (1) 12.

Moore, B. C. J., Peters, R. W. and Glasberg, B. R. 1993. Detection of Temporal Gaps in Sinusoids: Effects of Frequency and Level. **The Journal of the Acoustical Society of America**, 93, (3) 1563.

Mottola, R. M. 2006. Calculating Fret Positions. Available online at: www.liutaiomottola.com/formulae/fret.htm. Accessed: June 2008.

Naofumi Aoki, S. T., Eiichi Kishimoto, Seiki Yasuda, and Mutsuroh Iwakoshi 2004. Capturing Guitar Fingering by Photo-Reflector Technique. Joint Baltic-Nordic Acoustics Meeting Mariehamn,Åland. Available online at: C:\Documents and Settings\lcostalonga\My Documents\Macbook\Documents\Bibliography\EndNote Library\PDF Files\Aoki2004.pdf.

Norman, D. A. 1981. Categorization of Action Slips. **Psychological Review**, 88, (1) 1-15.

NothernDigital. 2009. Research Grade Motion Capture. Available online at: http://www.ndigital.com/lifesciences/index.php. Accessed: September 2009.

Palmer, C. 1989. Mapping Musical Thought to Musical Performance. **Journal of Experimental Psychology: Human Perception and Performance**, 15, (12) 331-346.

Palmer, C. 1992. The Role of Interpretive Preferences in Music Performance. In: Jones, M.R.H., Susan (Ed). **Cognitive Bases of Musical Communication**. American Psychological Association, Washington, DC, US. pp 249-262.

Palmer, C. 1997. Music Performance. **Annual Review of Psychology**, 48, (1) 115-138.

Palmer, C. and Van de Sande, C. 1993. Units of Knowledge in Music Performance. **Journal of Experimental Psychology: Learning Memory and Cognition**, 19, (2) 457-470.

Park, T. H. 2009. An Interview with Max Mathews. **Computer Music Journal**, 33, (3) 9-22.

Parlitz, D., Peschel, T. and Altenmuller, E. 1998. Assessment of Dynamic Finger Forces in Pianists: Effects of Training and Expertise. **Journal of Biomechanics**, 31, (11) 1063-1067.

Parncutt, R. 1994. A Perceptual Model of Pulse Salience and Metrical Accent in Musical Rhythms. **Music Perception**, 11, (4) 409-409.

Parncutt, R. 1997. Modeling Piano Performance: Physics and Cognition of a Virtual Pianist. International Computer Music Conference. . San Francisco, California, US.

Parncutt, R., Sloboda, J. A., Clarke, E. F., Raekallio, M. and Desain, P. 1997. An Ergonomic Model of Keyboard Fingering for Melodic Fragments. **Music Perception**, 14, (4) 341-382.

Penn, I. W., Chuang, T. Y., Chan, R. C. and Hsu, T. C. 1999. Emg Power Spectrum Analysis of First Dorsal Interosseous Muscle in Pianists. **Medicine and Science in Sports and Exercise**, 31, (12) 1834-1838.

Pennycook, B. W. 1985. Computer-Music Interfaces - a Survey. *Computing Surveys*, 17, (2) 267-289.

Perkell, J. S. and Klatt, D. H. 1986. **Invariance and Variability in Speech Processes**. Lawrence Erlbaum Assoc Inc, Hillsdale, NJ, England.

Pheasant, S. and Haslegrave, C. M. 2006. **Bodyspace: Anthropometry, Ergonomics, and the Design of Work**. CRC Press, Boca Raton, Florida, US.

Pisoni, D. B. 1977. Identification and Discrimination of the Relative Onset Time of Two Component Tones: Implications for Voicing Perception in Stops. **The Journal of the Acoustical Society of America**, 61, (5) 1352-1361.

Poepel, C. 2004a. **Synthesized Strings for String Players**. New interfaces for musical expression. Hamamatsu, Shizuoka, Japan National University of Singapore pp 150-153.

Poepel, C. 2004b. Synthesized Strings for String Players. **Proceedings of the 2004 conference on New interfaces for musical expression**, 150--153.

Poepel, C. 2005. On Interface Expressivity: A Player-Based Study. **Proceedings of the 2005 conference on New interfaces for musical expression**, 228--231.

Polansky, L. and Rosenboom, D. 1985. **Hmsl a Real-Time Environment for Formal, Perceptual and Compositional Experimentation**. International Computer Music Conference. Den Haag, Netherlands. Royal Conservatory of Music. pp 243-250.

Polansky, L., Rosenboom, D. and Burk, P. 1987. **Hmsl: Overview (Version 3.1) and Notes on Intelligent Instrument Design**. ICMC.

Povel, D. J. and Essens, P. 1985. Perception of Temporal Patterns. **Music Perception**, 2, (4) 411-440.

Puckette, M. 2002. Max at Seventeen. **Computer Music Journal**, 26, (4) 31-43.

Radicioni, D., Anselma, L. and Lombardo, V. 2004. A Segmentation--Based Prototype to Compute String Instruments Fingering. **Proceedings of the Conference on Interdisciplinary Musicology, Graz**,

Radicioni, D. and Lombardo, V. 2005. **Guitar Fingering for Music Performance**. International Computer Music Conference. Spain. 2005. pp 527-530.

Radicioni, D. P. and Lombardo, V. Computational Modeling of Chord Fingering for String Instruments. **strings**, 40, 45--50.

Radisavljevic, A. and Driessen, P. 2004a. Path Difference Learning for Guitar Fingering Problem. **Proceedings of the International Computer Music Conference**,

Radisavljevic, A. and Driessen, P. 2004b. **Path Difference Learning for Guitar Fingering Problem**. International Computer Music Conference. Miami, Florida, US.

Rasmussen, J. 1986. **Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering**. Elsevier Science Inc. New York, NY, USA.

Reason, J. 1987. Generic Error-Modelling System (Gems): A Cognitive Framework for Locating Common Human Error Forms. In: J Rasmussen, K.D., J Lepla (Ed). **New Technology and Human Error**. Wiley & Sons Chichester, New Yourk. pp 63--83.

Repp, B. H. 1994. Patterns of Expressive Timing in Performances of a Beethoven Minuet by 19 Famous Pianists - Response. **Journal of the Acoustical Society of America**, 96, (2) 1179-1181.

Repp, B. H. 2006. Rate Limits of Sensorimotor Synchronization. **Advances in Cognitive Psychology**, 2, (3) 163-181.

Roads, C. 1985. Research in Music and Artificial-Intelligence. **Computing Surveys**, 17, (2) 163-190.

Rosenbaum, D. A. 1995. Planning Reaches Bby Evaluating Stored Postures. **Psychological Review**, 102, (1) 28-67.

Rosenbaum, D. A. 1996. From Cognition to Biomechanics and Back: The End-State Comfort Effect and the Middle-Is-Faster Effect. **Acta Psychologica**, 94, (1) 59-85.

Salvendy, G. 1987. **Handbook of Human Factors**. Wiley.

Sanders, M. S. and McCormick, E. J. 1993. **Human Factors in Engineering and Design**. McGraw-Hill Science/Engineering/Math.

Saunders, C., Hardoon, D. R., Shawe-Taylor, J. and Widmer, G. 2004. **Using String Kernels to Identify Famous Performers**

from Their Playing Style. Boulicaut, J.F., Esposito, F., Giannoti, F. and Pedreschi, D. (Eds). 15th European Conference on Machine Learning. Pisa, Italy. Sep 20-24. pp 384-395.

Sayegh, S. I. 1989. Fingering for String Instruments with the Optimum Path Paradigm. **Computer Music Journal**, 13, (3) 76-84.

Schmidt, R. T. and Toews, J. V. 1970. Grip Strength as Measured by the Jamar Dynamometer. **Archives of Physical Medicine and Rehabilitation**, 51, (6) 321.

Seashore, C. E. 1936. **Objective Analysis of Musical Performance**. University of Iowa Press, Iowa, US

Seewald, A. K. 2002. **How to Make Stacking Better and Faster While Also Taking Care of an Unknown Weakness**. Nineteenth International Conference on Machine Learning table Morgan Kaufmann Publishers Inc. San Francisco, CA, USA. pp 554-561.

Shaffer, L. H. 1981. Performances of Chopin, Bach, and Bartok - Studies in Motor Programming. **COGNITIVE PSYCHOLOGY**, 13, (3) 326-376.

Shaffer, L. H. and Todd, N. P. 1987. **The Interpretive Component in Musical Performance**. Action and Perception in Rhythm and Music. Stockholm: Royal Swedish Academy of Music. University Press. pp 258-270.

Shan, G. B. and Visentin, P. 2003. A Quantitative Three-Dimensional Analysis of Arm Kinematics in Violin Performance. **Medical Problems of Performing Artists**, 18, (1) 3-10.

Shepard, R. N. 2002. Perceptual-Cognitive Universals as Reflections of the World. **Behavioral and Brain Sciences**, 24, (04) 581-601.

Shock, N. W. 1962. The Physiology of Aging. In: JH., P. (Ed). **Scientific American**. pp 100.

Silverman, B. W. and Jones, M. C. 1989. E. Fix and Jl Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951). **International Statistical Review**, 57, (3) 233-238.

Singleton , W. T. 1972. **Introduction to Ergonomics**. WHO, Geneva.

Sloboda, J. 1982. Music Performance. In: P Fraisse, D.D. (Ed). **The Psychology of Music**. Academic Press New York, NY. pp 479–496.

Sloboda, J. A. 2000. Individual Differences in Music Performance. **Trends in Cognitive Sciences**, 4, (10) 397-403.

Statsoft. 2009. Bagging. Available online at: http://www.statsoft.com/textbook/stdatmin.html#bagging. Accessed: Setpember 2009.

Sundberg, J. 1980. On the Anatomy of the Retard - a Study of Timing in Music. **Journal of the Acoustical Society of America**, 68, (3) 772-779.

Sundberg, J. 2000. Four Years of Research on Music and Motion. **Journal of New Music Research**, 29, (3) 183-185.

Sundberg, J. 2003. Special Issue: Research in Music Performance. **Journal of New Music Research**, 32, (3) 237-237.

Sundberg, J., Askenfelt, A. and Frydén, L. 1983a. Musical Performance: A Synthesis-by-Rule Approach. **Computer Music Journal**, 7, (1) 37-43.

Sundberg, J., Fryden, L. and Askenfelt, A. 1983b. What Tells You the Player Is Musical?An Analysis-by-Synthesis Study of Music Performance. In: Sundberg, J. (Ed). **Studies of Music Performance**. Royal Swedish Academy of Music, Stockholm, Sweden. pp 61–75.

Swain, A. D., Guttmann, H. E. and others 1983. Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications. **NUREG/CR**, 1278,

Thompson, W. F., Dalla Bella, S. and Keller, P. E. 2006. Music Performance. **Advances in Cognitive Psychology**, 2, (2-3) 99--102.

Todd, N. 1985. A Model of Expressive Timing in Tonal Music. **Music Perception**, 3, (1) 33-58.

Todd, N. 1989a. A Computational Model of Rubato. **Contemporary Music Review**, 3, (1) 69-88.

Todd, N. 1989b. Towards a Cognitive Theory of Expression: The Performance and Perception of Rubato. **Contemporary Music Review**, 4, (1) 405-416.

Todd, N. P. M. 1992. The Dynamics of Dynamics - a Model of Musical Expression. **Journal of the Acoustical Society of America**, 91, (6) 3540-3550.

Todd, N. P. M. 1995a. The Kinematics of Musical Expression. **Journal of the Acoustical Society of America**, 97, (3) 1940-1949.

Todd, N. P. M. A. 1995b. The Kinematics of Musical Expression. **The Journal of the Acoustical Society of America**, 97, 1940.

Tolonen, T. 1998. **Model-Based Analysis and Resynthesis of Acoustic Guitar Tones**. University of Helsinki

Tuohy, D. and Potter, W. D. 2005a. **A Genetic Algorithm for the Automatic Generation of Playable Guitar Tablature**. International Computer Music Conference. Barcelona, Spain. pp 499–502.

Tuohy, D. and Potter, W. D. 2005b. **A Genetic Algorithm for the Automatic Generation of Playable Guitar Tablature**. ICMC. pp 499–502.

Tuohy, D. R. and Potter, W. D. A Genetic Algorithm for the Automatic Generation of Playable Guitar Tablature. **Proc. International Computer Music Conference**, 499--502.

Tuohy, D. R. P., W. D. 2006. **An Evolved Neural Network/Hc Hybrid for Tablature Creation in Ga-Based Guitar Arranging**. Proceeding of the International Computer Music Conference New Orleans, Louisiana.

Valimaki, V., Huopaniemi, J., Karjalainen, M. and Janosy, Z. 1996. Physical Modeling of Plucked String Instruments with Application to Real-Time Sound Synthesis. **Journal of the Audio Engineering Society**, 44, (5) 331-353.

Vercoe, B. 1986. Csound: A Manual for the Audio Processing System and Supporting Programs. Program Documentation. Cambridge, Massachusetts: MIT Media Lab. Available online at: http://webland.panservice.it/musica/pavan/cspocman.pdf.

von Hornbostel, E. M. and Sachs, C. 1961. **Classification of Musical Instruments: Translated from the Original German by Anthony Baines and Klaus P. Wachsmann**. Galpin Society. 3-29 pp.

Wargo, M. J. 1967. Human Operator Response Speed, Frequency, and Flexibility: A Review and Analysis. **Human factors**, 9, (3) 221.

Weka. 2009. Use Weka in Your Java Code. Available online at: http://weka.wikispaces.com/Use+WEKA+in+your+Java+code. Accessed: September 2009.

Wickens, C. D. and Hollands, J. G. 2000. **Engineering Psychology and Human Performance (3rd)**. New Jersey: Prentice Hall.

Wickens, C. D., Lee, J., Liu, Y. D. and Gordon-Becker, S. 2003. **Introduction to Human Factors Engineering**. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Wickiewicz, T. L., Roy, R. R., Powell, P. L. and Edgerton, V. R. 1983. Muscle Architecture of the Human Lower Limb. **Clinical Orthopaedics and Related Research**, 179, (1) 275-283.

Widmer, G. 2003. Discovering Simple Rules in Complex Data: A Meta-Learning Algorithm and Some Surprising Musical Discoveries. **Artificial Intelligence**, 146, (2) 129-148.

Widmer, G. 2004. Computational Models of Expressive Music Performance: The State of the Art. **Journal of New Music Research**, 33, (3) 203-216.

Widmer, G., Dixon, S., Goebl, W., Pampalk, E. and Tobudic, A. 2003. In Search of the Horowitz Factor. **Ai Magazine**, 24, (3) 111-130.

Wing, A. M., Haggard, P. and Flanagan, J. R. 1996. **Hand and Brain: The Neurophysiology and Psychology of Hand Movements**. Academic Press San Diego.

Witten, I. H. and Frank, E. 2002. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. **ACM SIGMOD Record**, 31, (1) 76--77.

Wohlfart, B. and Edman, K. A. 1994. Rectangular Hyperbola Fitted to Muscle Force-Velocity Data Using Three-Dimensional Regression Analysis. **Experimental physiology**, 79, (2) 235-239.

Index

C

capacity demand
Central Nervous System
chord shape
classical guitar style
classical guitar techniques
Classification (Supervised) Learning
Clustering
cognitive models
cognitive theories
Common Music Notation
computer models
computer music programming
computer scientists
computer-modelling
contralateral activation
Correlation Coefficient
cross-section area

D

data mining
Decision Table
degree of freedom
deviation
dynamics

E

embodied
emotions
Endurance
ENP – Expressive Notation Package
ergonomic model
ergonomics

injuries
Instance-based (IBK)
inter-fret space
internal clocks
internal time-keeper clock
interonset intervals
intonation
IRCAN Ethersense Interface
isometric force
isometric strength

J
Jebsen Test of Hand Function
just noticeable difference

K
Kinematic Models
kinematics
KTH model
KTH performance rule system

L
lapses
left-hand
listening experiment
luthier

M
machine learning
mathematical modelling
Max/MSP
Maximum Practical Span
Mean Absolute Error

V
vibrato
virtuoso

W
Weka
white noise
workload

Y
Yamaha EZ-AG

GRAÚNA EDUFES