BIOLOGICALLY INSPIRED
ROBOTICS GROUP (BIRG)

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

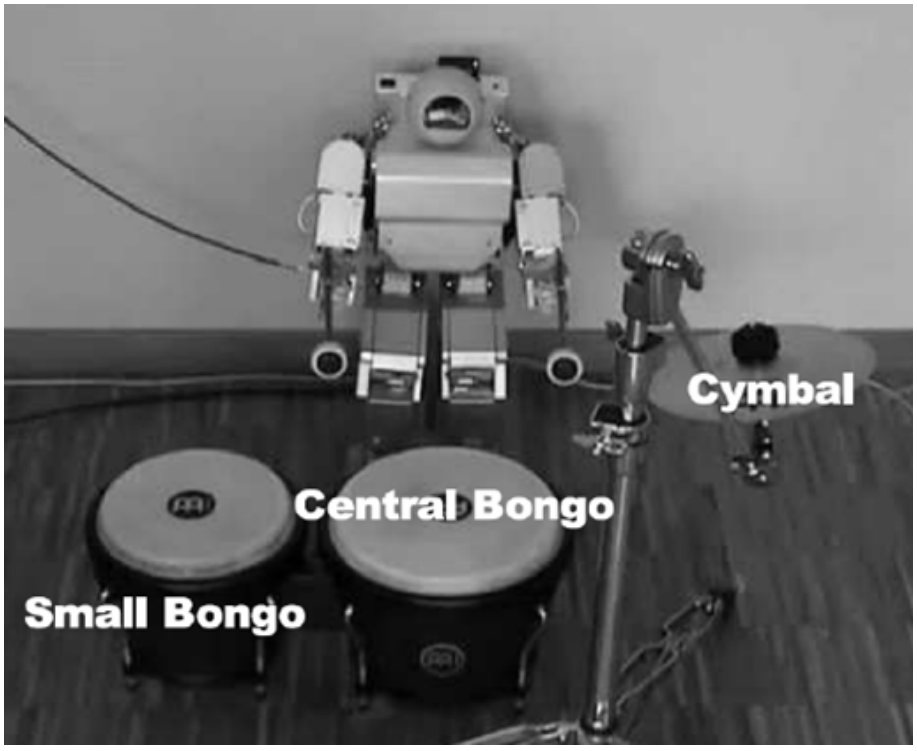
Winter 2007-2008, IC-BIRG, Project 393

Start: 17.09.2007
Finish: 15.02.2008

Faculty of Microengineering

Synchronization of movements of a real humanoid robot with music

Aïsha Hitz



Professor:	Auke Jan Ijspeert
Assistants:	Sarah Degallier, Ludovic Righetti

PROJECT 393:
Synchronization of Movements of a Real Humanoid Robot with Music

The goal of this project is to design a controller based on adaptive oscillators in order to synchronize the movements of a robot with music. A first study of synchronization of nonlinear oscillators and adaptation of frequency will be done and an implementation on the real Hoap2 humanoid robot will conclude the project.

The principal aim of the project is to develop an application of the concept of adaptive frequency oscillators. The idea is to use the existing drumming controller and to extend it to the synchronization with an external source of music.

The three main tasks are:

1. To understand the concepts of synchronization and of adaptive frequency oscillators. A theoretical part on those subjects should appear in the project.

2. To understand some of the techniques available for tempo detection, to present them and to discuss the feasibility of using adaptive oscillators instead.

3. To develop a controller for detecting the tempo using adaptive oscillators and to implement it on the robot Hoap2.

Already done:

- Development of tools to do animations with sound in Webots (P.Amstutz)
- Offline detection of the tempo using standard methods (P.Amstutz)

References:

S. H. Strogatz – an introduction to nonlinear systems
L. Righetti – adaptive oscillators
S. Degallier – drumming controller
E. D. Scheirer – an algorithm for tempo detection
P. Amstutz – semester project (see webpage: http://birg.epfl.ch/page65305.html)

SYNCHRONIZATION OF MOVEMENTS OF A REAL HUMANOID ROBOT WITH MUSIC

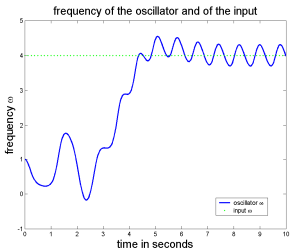
Aïsha Hitz, Faculty of Microengineering

Assistants: *Sarah Degallier, Ludovic Righetti*

Professor: *Auke Jan Ijspeert*

How do people synchronize with music? When listening to music, people intuitively detect a tempo and adapt their movement to the rhythm of music. They are then strongly influenced by an external signal that is the music. This behavior could be imitated using oscillators, dynamical systems that have the capability to synchronize to an external signal. Music could, in fact, be described as a periodic signal characterized by its frequency and its phase. As the frequency represents the tempo, the phase symbolizes the beats that are detected by people, when tapping along the rhythm of a song. Like people, oscillators should then adapt their behavior to the music, by modifying their frequency and their phase.

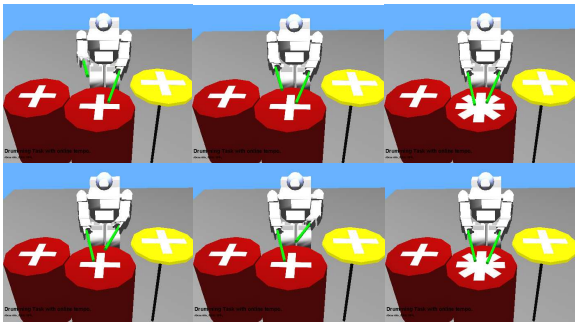
The purpose of this Master project was to implement adaptive frequency oscillators, dynamical systems which can adapt their frequency to the tempo of any music and to transmit it to a drumming controller of a humanoid robot. This last step helped to recreate visually a human-like behavior on a robot when listening to music.



Frequency adaptation of the oscillators.

The advantage of these adaptive frequency oscillators is their continuous adaptation. As long as there is an external signal applied to them, they adapt their frequency. The tempo detection can then be implemented in real-time, at the same time as music plays. It can support any changes in tempo and be robust to irregular beat, i.e. beats that are divided in different spaced meters and which can create perturbation.

The adaptive oscillators method developed in this project needed onset detection. An envelope extractor which emphasizes the sudden changes in sound, i.e. the moment of beat pulses, became the basis of the music pre-processing. Moreover, the efficiency of this method depends on the initializations of the parameters and of the initial conditions of the oscillators. Thus, if these are well fixed, the method allows a tempo detection of any kind of music, from simple impulse functions to electronic music, or from classical music to dance music.



The humanoid robot with its instrument.

As the study showed, there are many different aspects that are linked to tempo detection. Coordinated periodic movements as intuitively produced by human beings are not an easy task to be recreated, even if oscillators were good models to be used.

Some of the simulations done in Webots can be found on the website http://birg.epfl.ch/page67315.html.

Contents

1 INTRODUCTION **7**

2 ADAPTIVE FREQUENCY OSCILLATORS **9**
2.1 Oscillators and their synchronization 9
2.1.1 Phase locking and frequency entrainment 10
2.1.2 Coupled oscillators 12
2.2 Adaptive frequency oscillators 14
2.2.1 The Hopf oscillators 15
2.2.2 Adding adaptation 17
2.2.3 The parameters of the system 19
2.2.4 The ratio $\frac{\epsilon}{r}$ 21

3 MUSIC AND ITS ANALYSIS **26**
3.1 Some definitions 26
3.1.1 Music 26
3.1.2 Rhythm and meter 27
3.1.3 Pulse and beat 27
3.1.4 Tempo 28
3.2 Related work on music analysis 28
3.2.1 Scheirer's method 28
3.2.2 Shiu's method 29
3.2.3 Large's method 30
3.3 Method using oscillators 31

4 TEMPO DETECTION **33**
4.1 Music as an external periodic input 33
4.1.1 Mono- and stereophonic sound 33
4.1.2 Downsampling 34
4.1.3 The five tested songs 35
4.1.4 Repetition 35
4.2 Adaptation applied directly on music 37
4.3 The steps of Scheirer's method 38
4.3.1 Step1: Filterbank 38
4.3.2 Step2: Envelope extractor 38
4.3.3 Step3: Differentiator and half-wave rectifier 40
4.3.4 Step4: Comb-filters 40
4.4 Adaptation after pre-processing 41
4.4.1 The three pre-processing steps 43

4.4.2 First results 44
4.5 Adaptation time 46
4.6 Adaptation results 48
4.7 Existing methods 51
4.7.1 Compared to Scheirer's method 51
4.7.2 Compared to Shiu's method 51

5 SIMULATION **53**
5.1 Adaptation in real-time 53
5.1.1 Parallel computing 53
5.1.2 Threads 54
5.1.3 Implementation 55
5.2 The drumming task 56
5.2.1 Hitting the drum 58
5.2.2 Frequency limitation 60
5.3 Results on the robot 61
5.3.1 Oscillators behavior 62

6 CONCLUSION **63**

BIBLIOGRAPHY **65**

Chapter 1 INTRODUCTION

How do people synchronize with music? This question was asked in many research papers these past few years. When listening to music, people intuitively detect a certain tempo and move their body to the rhythm of the song. Would you actually had imagined that beat perception could be about predicting the future? When thinking about it, at best, people tap a rhythm considering the past beats and thinking that when they are going to tap the next time, there will be another beat happening. In [7], this was called "beat prediction"; it is possible because of the periodicity of beats. Interestingly, even when the beat stops, people continue to tap hoping that, next time, they would hear the beat again. After a while, if the beat does not appear when expected, they try to find another rhythm to tap along at periodic intervals of time. So people are strongly influenced by an external signal that is the music and especially its rhythm. People adapt their movement to the tempo of music. This behavior could then be imitated using oscillators, dynamical systems that have the capability to synchronize to an external signal. In this case, it is a periodic music signal characterized by its frequency and its phase. As the frequency represents the tempo, the phase symbolizes the beats that are detected by people, when tapping to the rhythm of a song. Like people, oscillators should then adapt their behavior to the music, by modifying their frequency and their phase.

The main purpose of this paper is to implement, for the humanoid robot Hoap2 by "Fujitsu", a controller based on adaptive frequency oscillators. These are able to synchronize with music, like people do when listening to music. Some projects related to this subject have already been done: an off-line tempo detection using standard methods and animations of the humanoid robot with music have been successfully implemented in Webots by P. Amstutz [1]. A drumming controller for the real humanoid robot has also been implemented by S. Degallier and explained in [4]. Adaptive frequency oscillators were developed by L. Righetti [14], dynamical systems which can adapt their frequency to an external input signal. So the goal of this project was to use part of these three projects to test a new tempo detection method. More specifically, the purpose was to find the "downbeats" of any song with these adaptive frequency oscillators and to transmit the music tempo to the drumming controller written by S. Degallier. This last step helped to visually recreate a robot with human-like behavior when listening to music.

The initial part of this paper consists in studying oscillators and especially their synchronization. Music theory was the subject of chapter 3, explaining tempo detection. Different techniques were analyzed in this chapter and a new method using oscillators is then presented. After testing the feasibility of using

the adaptive frequency oscillators for tempo detection, some general results are described, in chapter 4. An extension of the drumming controller on the humanoid robot Hoap2 was then considered in simulation. It is implemented in real-time, at the same time as music is playing. The different steps to obtain this on-line adaptation are explained in chapter 5, before showing some results in Webots.

Chapter 2 ADAPTIVE FREQUENCY OSCILLATORS

A lot of processes happening in nature are cyclic. The most obvious example is the day-night cycle. When scientists tried to describe them, they had to develop some specific models. Oscillators are a mathematical concept that could be used for such periodical behaviors. These dynamical systems are defined by their limit cycles and are mainly characterized by their capability to synchronize to periodical input signal. After a theoretical introduction and explanations of some specific oscillators behaviors, as frequency entrainment and phase locking, an example of such a system is described: the Hopf oscillator. The adaptive frequency oscillators are explained, based on this last system,.

2.1 OSCILLATORS AND THEIR SYNCHRONIZATION

Some of the motion types of dynamical systems give rise to very simple attractors, such as points and isolated closed curves called limit cycles. This cycle is a closed trajectory in the phase space and neighboring trajectories converge to it after a transient time. When they approach the limit cycle as time tends to infinity, it is considered as asymptotically stable or attractive. An oscillator can then be equally called as a stable limit cycle system.

The Van der Pol oscillator is an example of such a stable oscillator. Its dynamic equation, described with one state variable, is as:

$$\ddot{x} + \epsilon \left(x^2 - 1\right) \dot{x} + x = 0 \tag{2.1}$$

where ϵ controls the degree of nonlinearity of the system. In figure 2.1, the value of this parameter was fixed first equal to $\epsilon = 0.2$, then to $\epsilon = 1$ and finally to $\epsilon = 3$. It can be noticed that all three trajectories tended to a different limit cycle, when only changing the value of ϵ. It is a parameter influencing the shape of the limit cycle.

Modifying the initial conditions of a stable oscillator does not change its limit cycle. Since the Van der Pol oscillator has a stable limit cycle, this cycle is independent of the initial conditions. It is shown in figure 2.2: when changing them, all produced trajectories converge to the same limit cycle. The

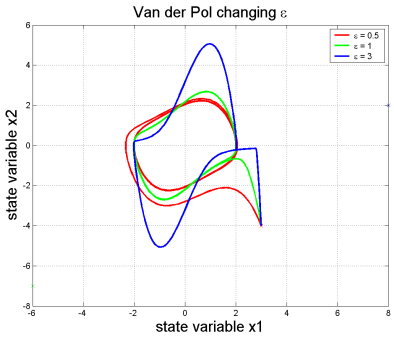


Figure 2.1: *different limit cycles of the Van der Pol oscillator, produced when changing only the value of the parameter* ϵ.

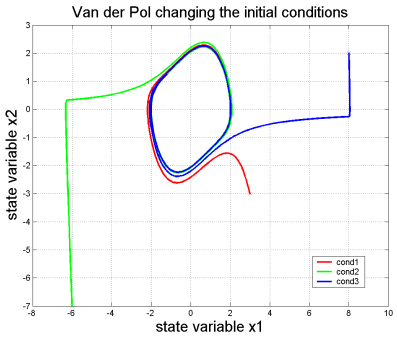


Figure 2.2: *different initial conditions are applied to the Van der Pol oscillator. Since it is a stable oscillator, all trajectories converge to a unique limit cycle.*

implemented initial conditions for this figure are quite different and are defined as:

$$\dot{x_1} = 3 \qquad x_1 = -3 \tag{2.2}$$

$$\dot{x_2} = -6 \qquad x_2 = -7 \tag{2.3}$$

$$\dot{x_3} = 8 \qquad x_3 = 2 \tag{2.4}$$

Stable limit cycles and so oscillators induce self sustained oscillations. If a small perturbation is introduced in the oscillator system so to get out of the limit cycle, it returns back to it after a certain time. In fact, the produced effects perturb the trajectory of the system for a certain time, but it returns to its stable limit cycle. This is the oscillator property of structural stability. In fact, stability theory is concerned about sensitivity of a system to small perturbations. A system is then said structurally stable if its dynamics do not change under small perturbation.

2.1.1 PHASE LOCKING AND FREQUENCY ENTRAINMENT

Synchronization can occur when an oscillator is influenced by an external input or when several oscillators are coupled together. It can be considered as an

adjustment of oscillations due to the oscillator interaction with an external input. Two different phenomena can describe synchronization: phase locking or frequency entrainment. Phase locking is reached when the oscillator phase ϕ can follow the phase of the external signal:

$$\phi = \omega t + const \tag{2.5}$$

When an oscillator is phase locked, its phase can also be defined, with its derivative:

$$\dot{\phi} = \omega = (const) \tag{2.6}$$

where ω is the external signal frequency. Its evolution in time is then a constant value. Frequency entrainment is reached when the observed frequency of the oscillator system is equal to the external signal frequency ω. The observed frequency has to be distinguished from the intrinsic frequency that is the initial frequency of the oscillator or the frequency when the oscillator is not perturbed. In fact, when an oscillator is influenced by an external signal its frequency changes to synchronize with the external signal frequency. It changes to a new frequency, that depends on both the intrinsic frequency and the external signal frequency. It is called the observed or the instantaneous frequency and represents the momentary rate of phase change. As for the unperturbed oscillator, the intrinsic frequency is a constant, for the perturbed system, the observed frequency becomes a function of time and is different from the intrinsic frequency.[13]

The phase ϕ is responsible of the oscillator synchronization. This variable describes motion along the limit cycle. It has the characteristic of being marginally stable. If an impulse of finite magnitude is added as an input to the system, the output will always grow. This output is said to be unbounded and no steady-state can be found. In this case, marginal stability explains the fact that the phase will always persist and so entrainment will always be possible, as long as the external signal stays. Some figures are shown later on in this chapter.

When a perturbation with a slightly different period as the oscillator one is added to this, the oscillator frequency changes and can become the one of the perturbation depending on the difference between those two frequencies. The mathematical explanation is given in the next section when presenting oscillators coupling. Frequency locking is then the moment when both frequencies, the one of the perturbation and the one of the oscillator, are equal. With another point of view, frequency entrainment is reached when the difference between the two phases remains constant. At this moment, the oscillator is entrained by the perturbation and the oscillations are synchronized with the frequency of the perturbation input. When both phases are the same so to get a zero phase difference, the signals oscillations are only distinguished by their amplitudes. This phenomenon is only possible when frequency entrainment was already reached.[13]

Applying a small perturbation, like an external periodic signal or another oscillating system, can produce large perturbations in the oscillator phase, but insignificant changes to its amplitude. So when adding an external input, effects on the phase ϕ moving along the limit cycle appear and can be analyzed. As shown in [14], in two dimensions, they can be divided in two main directions based on the limit cycle: the perpendicular effect that affects the radius of the limit cycle and that is damped out, and the tangential effect that influences the

phase of the oscillator. The domain where the perpendicular effects of the force are damped out is dependent of the coupling strength of the phase ϕ and the radius r. When the perturbation has disappeared, only the phase ϕ remains changed. This is due to its marginal stability, explained before. In contrast with this, the amplitude of the oscillations has a definite and limited value; when a perturbation changes the amplitude value, this tries to get its initial value back, thanks to an "equilibrium of energy influx and losses" as said in [14]. The phase ϕ point returns then to the limit cycle, so that the initial radius r is reached again; but it is time-delayed according to the initial oscillator phase. The resulting signals can then have very different amplitudes and waveforms, but they oscillate with the same frequency.

2.1.2 COUPLED OSCILLATORS

Synchronization is shown to be dependent on the frequency difference of the two systems coupled together. This is mathematically explained, when coupling two phase oscillators. Some tests were made to visualize their behavior.

The coupling equations are:

$$\dot{\phi_1} = \omega_1 + k \ \sin(\phi_2 - \phi_1) \tag{2.7}$$
$$\dot{\phi_2} = \omega_2 \tag{2.8}$$

where k represents the coupling strength. The initial phases and k were not changed and fixed at $\phi_1 = 0.1rad$, $\phi_2 = 0.5rad$ and $k = 3$. When looking at the two equations, only the second oscillator influences the first one, i.e. the first oscillator (equation 2.7) should synchronize with the second one (equation 2.8). But the two frequencies have to be close to each other to get entrained. The first oscillator, in this case, will have the tendency to decelerate or accelerate depending on the frequency difference, and so depending on the phase equation:

$$\phi_{diff} = \phi_2 - \phi_1 = const \tag{2.9}$$
$$\dot{\phi}_{diff} = \dot{\phi_2} - \dot{\phi_1} \quad with \quad \dot{\phi_1} = \omega_1 + k \ \sin(\phi_2 - \phi_1) \quad \dot{\phi_2} = \omega_2 \tag{2.10}$$
$$\dot{\phi}_{diff} = \omega_2 - \omega_1 + k \ \sin(\phi_2 - \phi_1) \tag{2.11}$$

When the oscillator is synchronized with the input, their phase difference is a constant and its derivative null. So the following equalities are obtained:

$$\dot{\phi}_{diff} = 0 = \omega_2 - \omega_1 + k \sin(\phi_2 - \phi_1) \tag{2.12}$$

The solution that is the possible phase difference becomes:

$$\frac{\omega_1 - \omega_2}{k} = \sin(\phi_2 - \phi_1) \tag{2.13}$$
$$\phi_{diff} = \arcsin(\frac{\omega_1 - \omega_2}{k}) \tag{2.14}$$
$$-1 < \frac{\omega_1 - \omega_2}{k} < 1 \tag{2.15}$$

It can be noticed that the synchronization depends not only on the frequency difference, but also on the parameter k, the coupling strength. Moreover, the frequency ratio must be smaller than 1 in an absolute value to allow convergence.

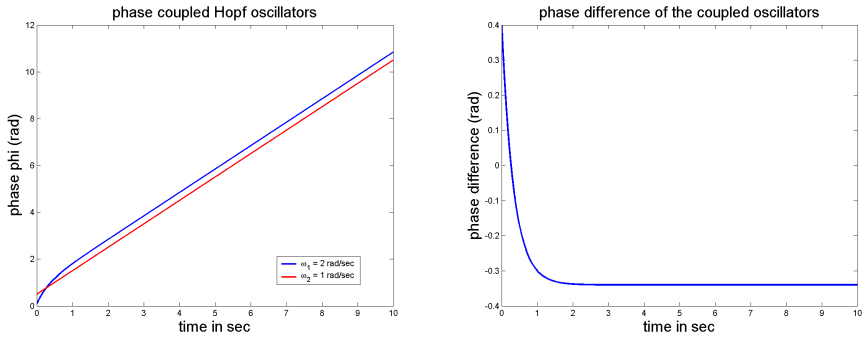


Figure 2.3: *the different phase plots (left) and the phase difference of two coupled oscillators (right) with different intrinsic frequency:* $\omega_1 = 2rad/s$ *and* $\omega_2 = 1rad/s$*. Entrainment is shown on both graphs.*

The basin of entrainment of the system is then defined as the maximum distance possible between the frequency of the oscillator and the frequency of the input to reach synchronization [14]. Inside of this basin of entrainment, convergence is guaranteed. Outside the basin, synchronization is no more possible, but the oscillator is still influenced by the coupled input. The value of the oscillations frequency should be situated in general between the initial oscillator frequency and the external input frequency.

The frequencies of the two phase oscillators were changed and some observations were done on the difference of their phase first and then on the evolution of their phase difference. For the first experiment, the oscillators frequencies were fixed at $\omega_1 = 2rad/s$ and at $\omega_2 = 1rad/s$. The frequency synchronization can be noticed on both plots, in figure 2.3. On the left one, when both lines are parallel, it can be seen that both phases increase at the same rate and so that the phase difference is a constant value. On the right of figure 2.3, when their phase difference becomes a horizontal line and shows a null derivative, synchronization is also reached. In this case, there is a negative phase shift. This means that the second oscillator is delayed; when referring to equation (2.9), ϕ_1 is bigger than ϕ_2. This can be deduced from figure 2.3, on the left: for a given time, ϕ_1 is always bigger than ϕ_2.

With the second experiment, in figure 2.4, the oscillators frequencies were fixed at $\omega_1 = 2rad/s$ and at $\omega_2 = 5rad/s$. The first oscillator again adapts its frequency to the value of the second one. In this case, there is a positive phase shift, showing that the first oscillator is delayed. The final value of the phase difference is higher than for the first experiment, but it is still a constant one. This is also shown in the left plot, with the two parallel lines.

For the last experiment, the oscillators frequencies were fixed at $\omega_1 = 2rad/s$ and at $\omega_2 = 6rad/s$. In this case, the frequency adaptation is not possible: the first oscillator is not able to reach the frequency value of the second oscillator. In fact, when replacing the initial values into equation (2.15), the following inequality is obtained:

$$\frac{\omega_1 - \omega_2}{k} = \frac{2-6}{3} = \frac{-4}{3} \quad < \quad -1 \tag{2.16}$$

and in this case, the frequency ratio is smaller than -1 and so outside of the

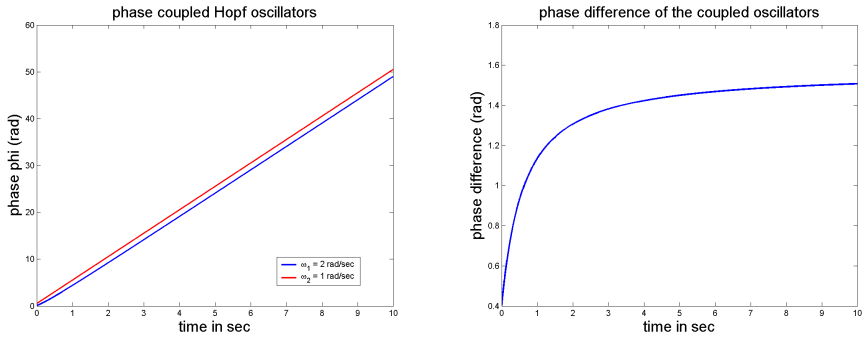


Figure 2.4: *phase plots (left) and phase difference of two coupled oscillators (right) with different intrinsic frequency:* $\omega_1 = 2rad/s$ *and* $\omega_2 = 5rad/s$*. Entrainment is also shown on both graphs.*

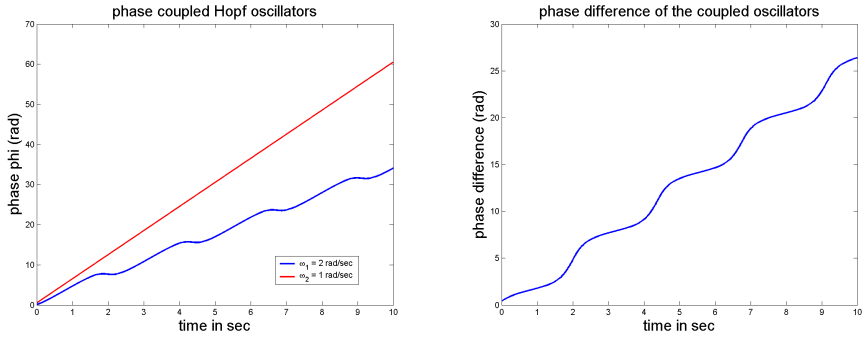


Figure 2.5: *phase plots (left) and phase difference of two coupled oscillators (right) with different intrinsic frequency:* $\omega_1 = 2rad/s$ *and* $\omega_2 = 6rad/s$*. Entrainment is not shown any more, but the right plot shows regularity in the increase of the phases.*

basin of entrainment.

On the right plot of figure 2.5, it can be noticed that the phase difference always varies, but with a certain regularity, i.e. a certain dependence. In fact, the phase difference derivative is an almost constant value (~ 4.6) and so the evolution of the phases increases regularly. It shows the influence of the second oscillator on the first one.

2.2 ADAPTIVE FREQUENCY OSCILLATORS

During synchronization, an adaptation of frequency can be done. In fact, the dynamical system can be modified such as it can slowly adapt its frequency to the external signal. This is done when using adaptive frequency oscillators: their frequency is described as a state variable and it changes over time. It introduces the idea of memory in the behavior of the oscillators [14]. Their equations are presented after characterizing the Hopf oscillator. Some tests are then done to illustrate their reaction to an external periodic signal.

2.2.1 THE HOPF OSCILLATORS

The Hopf oscillator is described with a differential equation system represented by two independent states. In Cartesian coordinates, it is defined as:

$$\dot{x} = \alpha\left(\mu - \left(x^2 + y^2\right)\right)x - \omega y \tag{2.17}$$

$$\dot{y} = \alpha\left(\mu - \left(x^2 + y^2\right)\right)y + \omega x \tag{2.18}$$

where μ controls the amplitude of the oscillations, α influences the time taken to reach the limit cycle and ω represents the intrinsic frequency of the Hopf oscillator. The first part of the equations gives a hint about the stability and the behavior of the system. In fact, μ is the energy parameter and depending on this value, two main behaviors of the oscillator can be observed. If $\mu < 0$, the system has a stable fixed point: the oscillations are damped out till $r = 0$ and it can be said that the energy is taken out of the system. If $\mu > 0$, the energy can be said to be added and self-sustained oscillations arises till reaching a stable limit cycle [8]. So, in this case, μ is fixed greater than 0 to assure the presence of oscillations. It represents the stable energy to reach and the term $\left(\mu - \left(x^2 + y^2\right)\right)$ becomes then positive or negative depending on the instantaneous state of the oscillator.

When writing these same equations with the corresponding polar coordinates:

$$x = r\cos(\phi) \qquad y = r\sin(\phi)$$

the system becomes:

$$\dot{r} = \alpha\left(\mu - r^2\right)r \tag{2.19}$$

$$\dot{\phi} = \omega \tag{2.20}$$

The evolutions of r and of ϕ in time are independent of each other. Moreover r tends to $\sqrt{\mu}$ when the limit cycle is reached and when the system is stable. This radius r becomes then an equilibrium point of the system. The stability is shown when the partial derivative of the function evaluated at this equilibrium point is always negative. Mathematically, in a system $\dot{x} = f(x)$, a fixed point or an equilibrium point of a system is defined as:

$$\dot{\overline{x}} = f(\overline{x}) = 0 \tag{2.21}$$

And the stability of the system is evaluated like:

$$\left.\frac{\partial f}{\partial x}\right|_{\overline{x}} \begin{cases} < 0 & \text{system is stable} \\ > 0 & \text{system is unstable} \\ = 0 & \text{system stability is not predictable} \end{cases} \tag{2.22}$$

In this case, since μ is a positive value so to have oscillations, the stability is assured:

$$\frac{\partial}{\partial r}\left(\left(\left(\mu - r^2\right)r\right)\right) = \left(\mu - r^2\right) - 2\,r^2 = \mu - 3\,r^2 \tag{2.23}$$

$$r = \sqrt{\mu} \quad \rightarrow \quad \mu - 3\,r^2 = -2\,\mu < 0 \tag{2.24}$$

This stability is important when the oscillator is perturbed as explained in section 2.1.1.

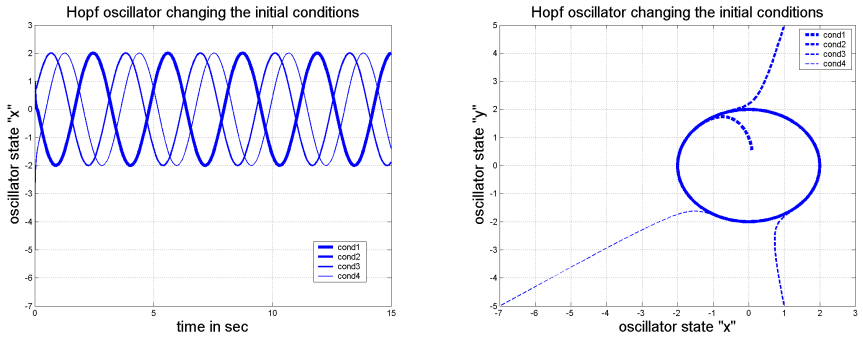


Figure 2.6: *representation of the Hopf oscillator when changing the initial conditions. The phase state shows a phase difference but a same frequency for all four trajectories (left) and they all converge to the same limit cycle (right).*

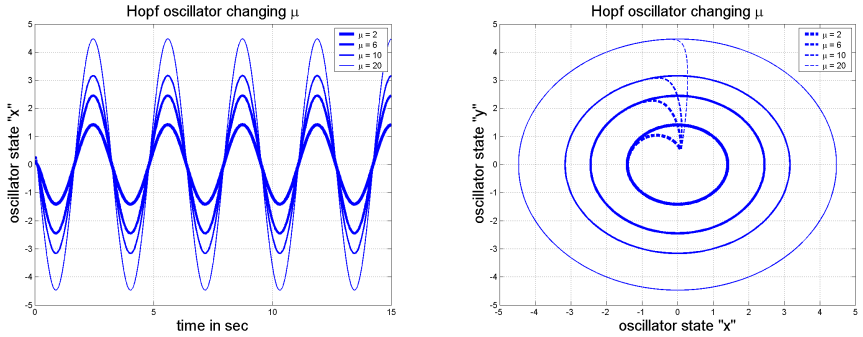


Figure 2.7: *representation of the Hopf oscillator when changing the parameter μ. It modifies the amplitude of the oscillations (left) and the size of the limit cycle (right).*

Looking at the Hopf oscillator equations, there are different parameters that can influence its behavior: the frequency ω, the initial conditions, μ and finally α. Some experiments are shown now when changing them. First of all, when referring to equations (2.19) and (2.20), in polar coordinates, it can be noticed that both the radius r and the angle ϕ, i.e. the frequency ω, are independent of each other. So when modifying the intrinsic frequency ω, the limit cycle is not changed.

In figure 2.6, a simple Hopf oscillator is drawn changing only the initial conditions of x and of y. All oscillations vibrate with the same intrinsic frequency and all resulting trajectories converge to the same limit cycle of the Hopf oscillator. The initial conditions have no effect on either the limit cycle or the frequency. However it can be noticed that even if the frequency is the same, there is a phase shift between them.

Then μ is modified and the results are shown in figure 2.7. This parameter influences the amplitude of the oscillations and the size of the resulting limit cycle of the oscillator. In fact, the higher the value of this variable, the bigger the amplitude and the larger the limit cycle of the oscillations.

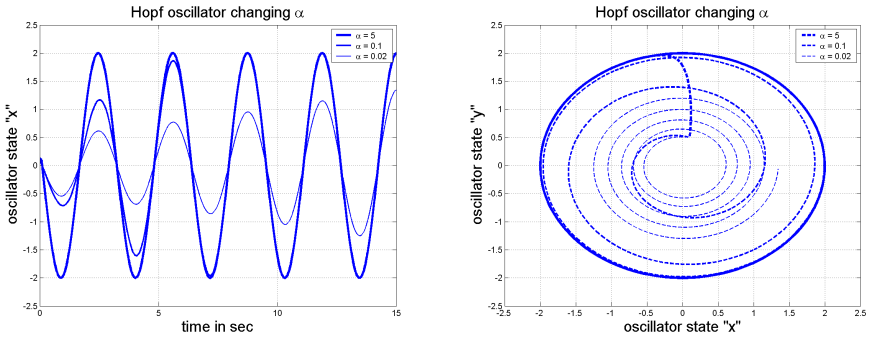


Figure 2.8: *representation of the Hopf oscillator when changing the parameter α. It has an influence on the time needed to reach the oscillations amplitudes (left) and on the time taken to converge to the limit cycle (right).*

The last parameter to test is α. For the oscillations, on the left in figure 2.8, it can be noticed that it has an important influence on the time taken to obtain their final amplitude; the smaller the parameter, the longer the time. On the right in figure 2.8, changing α modifies also the time to reach the limit cycle; it can be seen that the smaller the parameter α, the longer the time spent by the trajectory to reach the limit cycle.

To sum up the relevant points, it can be said that:

- modifying the frequency ω has no effect on the limit cycle.
- the initial conditions have no effect on either the limit cycle or the frequency. Only the instantaneous phase is influenced.
- the higher the value of the parameter μ, the bigger the amplitude and the larger the limit cycle of the oscillations.
- the smaller the parameter α, the longer the time to obtain the final amplitude of the oscillations and the longer the time spent by the trajectory to reach the limit cycle.

2.2.2 ADDING ADAPTATION

Adaptation is then added to the Hopf oscillator thanks to a third equation describing the evolution of the oscillator frequency: it evolves toward the frequency of the external input signal. For oscillators in general, the synchronization is lost as soon as the external signal has disappeared. With the frequency adaptation, when stopping the influence of the external signal, the oscillators keep the last frequency value at which they were oscillating with the presence of this signal. The information introduced through the external input is memorized in the system thanks to a learning process as explained in [2].

The equation of a classic oscillator perturbed by an external periodic input $F(t)$ is:

$$\dot{x} = f_1(x, y, w) + \epsilon F(t) \quad (2.25)$$

$$\dot{y} = f_2(x, y, w) \quad (2.26)$$

where x and y are the state variables of the oscillator, ω characterizes its intrinsic frequency and ϵ the coupling strength. When transforming this frequency ω into a state variable so that it can be included in the oscillator system, the additional equation becomes:

$$\dot{\omega} = -\epsilon \; F(t) \frac{y}{\sqrt{x^2 + y^2}} \tag{2.27}$$

The oscillator can then adapt its frequency to the frequency of any external periodic signal $F(t)$. In [14], it was demonstrated that such oscillators could even adapt their frequencies to pseudo-periodic and noisy signals. Equation (2.27) can be used with all kind of oscillators, like phase oscillators that consider only the phase and neglects the radius, the isochronous oscillators, like the Hopf oscillator, the harmonic oscillators with a harmonic limit cycle, etc. (see [3] for more details about the different oscillators and their characteristics).

For this project, the Hopf oscillator is used. When adding this frequency adaptation to the Hopf oscillator equations, the following system is obtained, with the external signal $F(t)$:

$$\dot{x} = \left(\mu - \left(x^2 + y^2\right)\right) x - \omega y \tag{2.28}$$

$$\dot{y} = \left(\mu - \left(x^2 + y^2\right)\right) y + \omega x \tag{2.29}$$

$$\dot{\omega} = -\epsilon \; F(t) \frac{y}{\sqrt{x^2 + y^2}} \tag{2.30}$$

Looking at the same system in the polar coordinates, the following equations are obtained:

$$\dot{r} = \left(\mu - r^2\right) r + \epsilon \; F(t) \cos(\phi) \tag{2.31}$$

$$\dot{\phi} = \omega - \frac{\epsilon}{r} \; F(t) \sin(\phi) \tag{2.32}$$

$$\dot{\omega} = -\epsilon \; F(t) \sin(\phi) \tag{2.33}$$

When the input $F(t)$ is zero, the system has an asymptotically stable harmonic limit cycle, with frequency ω and radius $r = \sqrt{\mu}$; the unperturbed Hopf oscillator is obtained. The structural stability of this oscillator assures that its qualitative behavior does not change when small perturbations are applied around its limit cycle. In [14], it was proven that the learning rule of the Hopf oscillator always converges to the frequency of the external signal. The final adapted frequency is actually an oscillation around the frequency of the external input. It has to be noticed that if the external signal is a complex periodic signal, the oscillator converges to one of the signal frequencies. The amplitude of the oscillations is much smaller than the value of ϵ; actually it is of the order of $\frac{\epsilon}{4\omega_F}$ as demonstrated by the same author. The oscillations pattern can take various shapes, but for each system, this pattern always becomes a periodic function after synchronization. It depends on the difference of both frequencies, the oscillator one and the external signal one, and on the amplitude of the external input.

After testing the Hopf oscillator alone, some other experiments were performed with the frequency adaptation. An adaptive frequency oscillator was perturbed with an external signal and changes of its parameters were implemented. The oscillator reactions are explained in the next sections, with a particular attention directed to the effects of ϵ, the coupling strength.

2.2.3 THE PARAMETERS OF THE SYSTEM

When looking at the adaptive frequency oscillators equations, the same parameters as for the Hopf oscillator appear: the frequency ω, the phase ϕ and the radius r. Some first tests were made changing these parameters, when the external signal was initialized as a simple sine wave:

$$F(t) = \sin(\omega t) \tag{2.34}$$

The oscillators parameter were initialized at $\epsilon = 10$ and at $\mu = 9$. Adding adaptation does not change the analysis of phase plots; so that, the explanation given in section 2.1.2, for the phase oscillators are still applicable with adaptive frequency oscillators. Adaptation is shown on them when the phase evolution line becomes a periodic function, depending on the initial conditions of the system. In fact, the moment of adaptation can be defined as the point in time when the phase evolution pattern becomes constant.

When adaptation is reached, the phase derivative is constant and the frequency becomes a constant value, the one of the external input signal. In fact, the frequency evolution tends to zero, so that the oscillator frequency becomes the one of the external signal. Then, the phase evolution tends to the frequency ω. It is shown with these equations:

$$\dot{\omega} = -\epsilon\, F(t) \sin(\phi) \tag{2.35}$$

$$\dot{\phi} = \omega - \frac{\epsilon}{r}\, F(t) \sin(\phi) \tag{2.36}$$

Replacing equation (2.35) into equation (2.36), the following relation is obtained:

$$\dot{\phi} = \omega - \frac{\dot{\omega}}{r} \tag{2.37}$$

There is a dependence between the phase and the frequency evolutions of the oscillator, due to the equations of the system. Two different examples are shown in figure 2.9, where both behaviors, the one of the phase and the one of the frequency, can be observed. As the frequency monotonically increases or decreases, the phase shows also a monotonous evolution. On the left of figure 2.9, both the frequency and the phase grow monotonically, for example.

With the same experiments, the radius r as a function of time was plotted. Knowing that for the chosen oscillator the limit cycle radius was:

$$r = \sqrt{\mu} = \sqrt{4} = 2 \tag{2.38}$$

it can be noticed that in the presence of an external input, the radius is not constant any more; but it varies around its initial value. This is due to the structural stability of the Hopf oscillator, a characteristic presented in chapter 2.1. When adaptation is reached, the radius r oscillates, as said before, around a constant value. As shown in figure 2.10, the oscillations patterns are different, depending, among other things, on the initialization of the frequencies, i.e. the difference between the oscillator and the external input frequencies.

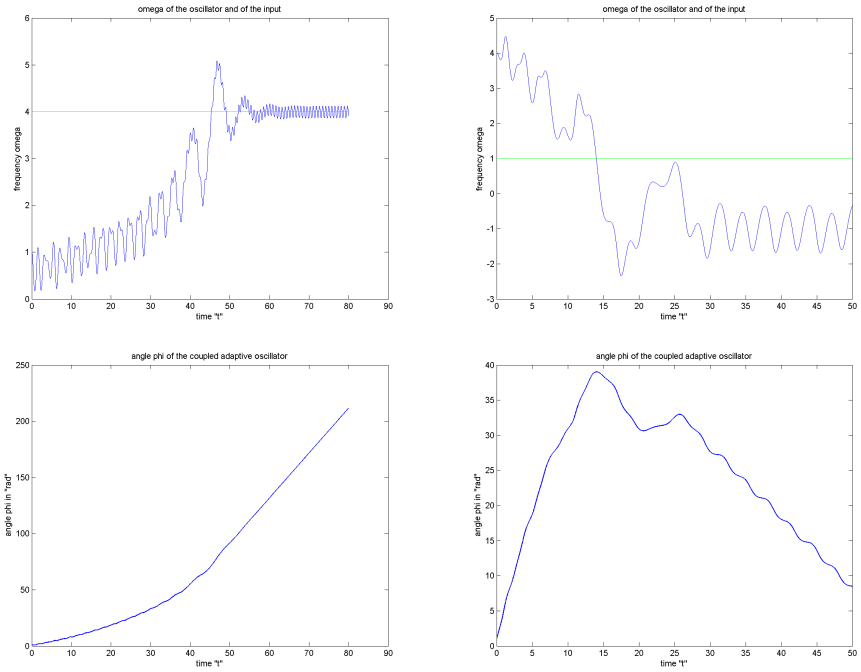


Figure 2.9: *the initial frequencies of the left experiment are, for the oscillator,* $\omega_1 = 4rad/s$ *and, for the input,* $\omega_2 = 1rad/s$*. For the right experiment, the values are* $\omega_1 = 1rad/s$ *and* $\omega_2 = 4rad/s$*. The evolution of the phase plot (bottom) and the frequency plot (top) can be compared: when the phase changes, the frequency is modified too.*

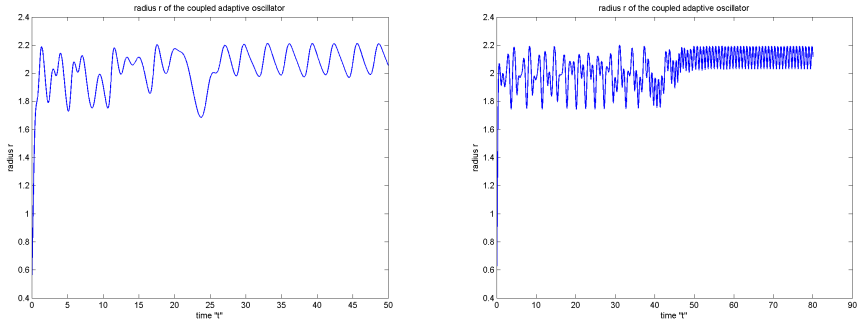


Figure 2.10: *the initial frequencies of the left experiment are,* $\omega_1 = 4rad/s$ *and* $\omega_2 = 1rad/s$*, and for the right one,* $\omega_1 = 1rad/s$ *and* $\omega_2 = 4rad/s$*. Both oscillations patterns are obviously different, but the constant value of the radius* $r = 2$ *is the same.*

When removing the influence of the external force on the radius, so that the equations of the system used are:

$$\dot{r} = \left(\mu - r^2\right) r \tag{2.39}$$
$$\dot{\phi} = \omega - \frac{\epsilon}{r} F(t) \sin(\phi) \tag{2.40}$$
$$\dot{\omega} = -\epsilon F(t) \sin(\phi) \tag{2.41}$$

there are no more oscillations around the constant radius r. It takes the constant value of $r = \sqrt{\mu}$.

Several experiments were done when changing the external input signal, described by a sum of two sine and two cosine waves with different amplitudes for each of them. When applying the same force on the oscillator with or without the influence on the radius, the final learned frequency of the oscillator was the same for all experiments. This can be seen with the results of two of them, where the input forces were:

$$F_1(t) = \cos(200t) + 4\cos(50t) + 7\sin(150t) + 10\sin(30t) \tag{2.42}$$
$$F_2(t) = 5\cos(50t) + 10\cos(200t) + \sin(70t) + 2\sin(30t) \tag{2.43}$$

The oscillators parameters were fixed at $\phi = 1rad$, $\omega = 100rad/sec$, $\epsilon = 10$ and $\mu = 9$. For the first experiment, on the upper part of figure 2.11, the oscillator adapts its frequency to the last sine wave frequency ($\omega = 30rad/s$) and for the second experiment, on the lower part of figure 2.12, it adapts the first cosine wave frequency ($\omega = 50rad/s$).

Removing the influence of the external input on the radius modifies also the phase evolution since the radius r influence is now a constant value. This comes directly from the phase equation of the system:

$$\dot{\phi} = \omega - \frac{\epsilon}{r} F(t) \sin(\phi) \tag{2.44}$$

This change makes the adaptation slower, but the learned frequency of the oscillator is then more precise, as the radius r is constant. This is demonstrated in figure 2.12, which is a zoom of the upper experiment shown in figure 2.11. Since its influence on the system convergence is linked to another parameter, that is ϵ, the influence of the external input on the radius can be removed without modifying the general behavior of the oscillators. This ϵ is an important parameter: it appears in both the phase and the frequency evolution. The next section will give some more details about that.

2.2.4 THE RATIO $\frac{\epsilon}{r}$

In general, when looking at the equations of the adaptive frequency oscillator, removing the influence of the external signal on the radius r, some first conclusions can be done:

$$\dot{r} = \left(\mu - r^2\right) r \tag{2.45}$$
$$\dot{\phi} = \omega - \frac{\epsilon}{r} F(t) \sin(\phi) \tag{2.46}$$
$$\dot{\omega} = -\epsilon F(t) \sin(\phi) \tag{2.47}$$

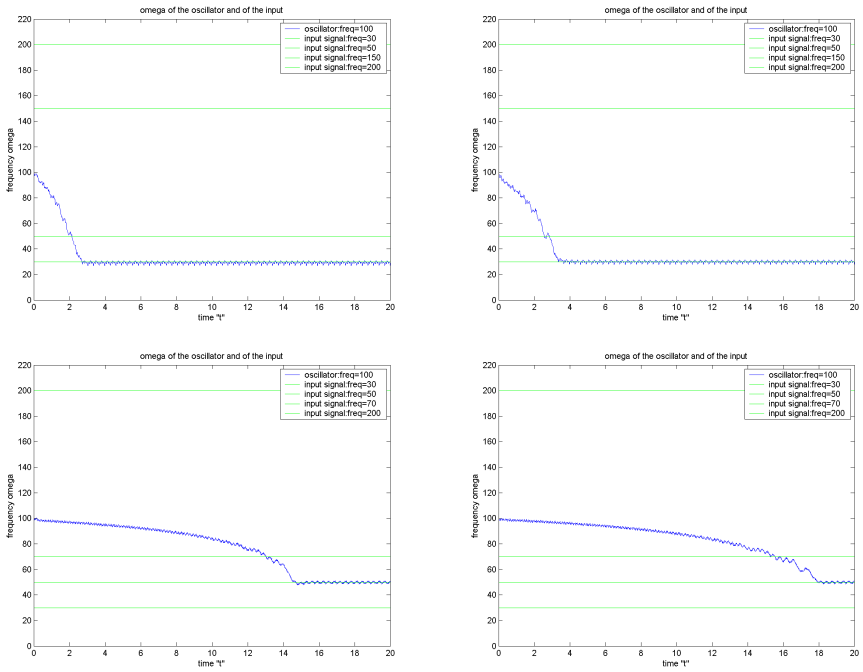


Figure 2.11: *two experiments differentiated only by the change of the external input signal are plotted here; one on the upper part and the other one on the lower part of the figure. In both cases, it can be noticed that with (left) or without (right) the influence of the input signal on the radius r, the oscillator adapts the same input signal frequency.*

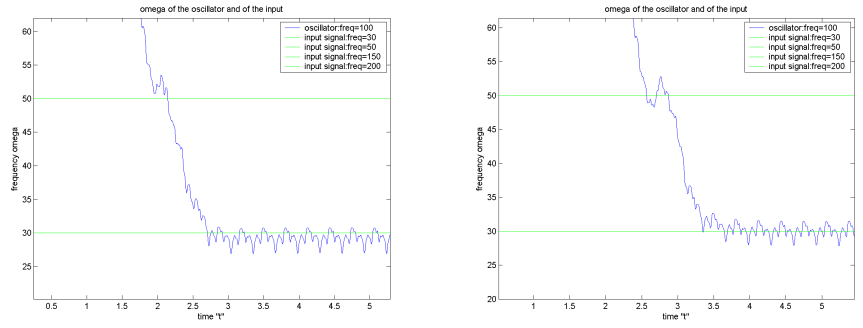


Figure 2.12: *the difference of the oscillator adapted frequency can be noticed when keeping (left) or removing (right) the influence of the external input signal on the radius r. In fact, without this influence, the adapted frequency becomes closer to the external input signal frequency.*

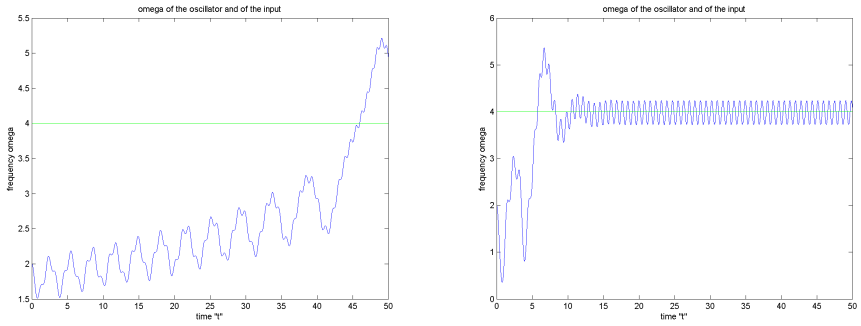


Figure 2.13: *with the frequencies of* $\omega_1 = 2rad/s$ *for the oscillator and* $\omega_2 = 4rad/s$ *for the input, the parameter* ϵ *is modified: on the left,* $\epsilon = 1$ *and on the right,* $\epsilon = 4$*. Since the time interval is kept the same, it can be noticed that the higher* ϵ*, the quicker the adaptation.*

Knowing that the implemented function that transforms the music into a vector of data returns only data amplitude smaller than 1 and that the sine function is always situated in a range of [-1, 1], the evolution of the oscillator frequency, described by equation (2.47) depends on the value of ϵ. It is actually proportional to it. This parameter should then not be too high, remembering that, to get adaptation, this last equation should tend to zero.

Some tests were then done when initializing the external force to a simple sine wave. The value of ϵ was changed, when fixing $\mu = 1$, to observe its influence on the adaptation. This parameter ϵ is actually responsible, for example, of the learning rate. The plots in figure 2.13 are obtained with $\epsilon = 1$ and $\epsilon = 4$. They show an important point: the higher ϵ, the quicker the adaptation.

This point is also relevant when perturbing the adaptive frequency oscillators with a function of impulses at a frequency of $2Hz$, in figure 2.14. When $\mu = 1$, ϵ is changed several times: $\epsilon = 5$, $\epsilon = 10$, $\epsilon = 20$, $\epsilon = 50$ and $\epsilon = 100$. It can be noticed that, in fact, the higher the parameter, the shorter the time to converge. At the same time, the adapted value becomes more precise. Thus, as ϵ gets higher, the oscillations around the adapted frequency increases. This is observed when $\epsilon = 100$. This second relevant point is related in [14]: since ϵ controls the adaptation rate of the system, the higher ϵ, the faster this rate gets and the higher the error of the adaptation becomes.

The sine wave frequency was then initialized at $\omega = 100rad/s$ and the oscillator started with a frequency of $\omega = 20rad/s$. Plots in figure 2.15 were obtained, when modifying again the value ϵ. It can be noticed that, if this value is too high, even after convergence, the oscillator frequency can loose track of the external signal frequency. On the other hand, if it is too small, the time of convergence become much longer.

For the Hopf oscillator system, the ratio $\frac{\epsilon}{r}$ appears in the phase evolution equation. Since the influence of the external input is removed for the radius r, this ratio becomes: $\frac{\epsilon}{\sqrt{\mu}}$. With the experiments shown in figures 2.13, 2.14 and 2.15, it can be concluded that a compromise has to be done, when fixing the value of ϵ: a compromise between high speed of convergence and precision

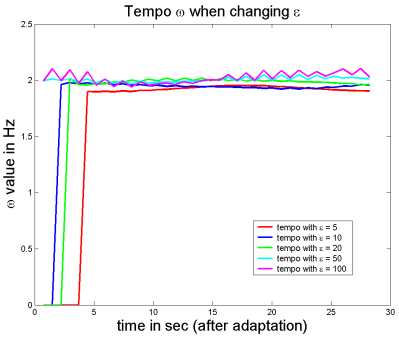


Figure 2.14: *the different adapted frequencies of an input with impulses appearing at a frequency of* $2Hz$*, when changing* ϵ*. As this parameter is increased, the adaptation takes more time and the value of the adapted frequency becomes closer to the input frequency, but more variable.*

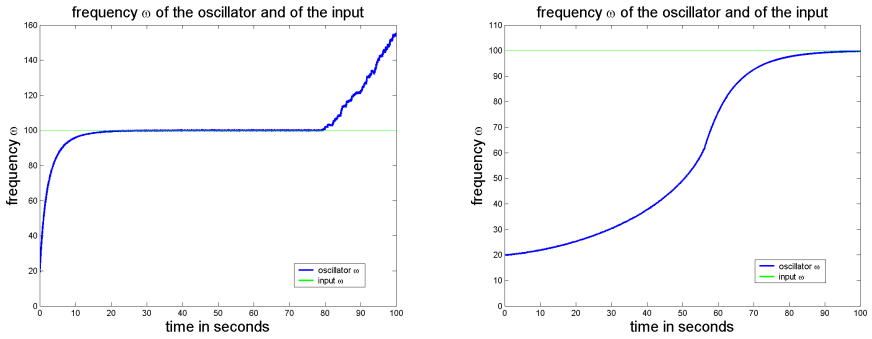


Figure 2.15: *adaptation of an oscillator with initial parameters* $\mu = 0.01$ *and* $\omega = 20rad/s$*, to an external sine wave of* $\omega = 100rad/s$*. When fixing the coupling strength too high,* $\epsilon = 500$ *(left), the oscillator frequency can loose adaptation; when fixing it too small,* $\epsilon = 10$ *(right), the time for convergence is increased.*

and stability for the adapted frequency. This initialization influences then also the ratio $\frac{\epsilon}{r}$. The smaller the ratio, the closer the oscillator frequency can get to the external frequency and the more precise this becomes. The higher the ratio, the quicker the adaptation and the further the initial oscillators frequency can be fixed from the external signal frequency. Thus, as the ratio increases, the average value of the adapted frequency changes and gets higher, like the amplitude of the oscillations around the adapted frequency. In fact, the ratio $\frac{\epsilon}{r}$ and its influence on the phase evolution increase. So the precision on the adapted frequency gets lost. If it has too much influence on the frequency evolution, this frequency derivative can not be annihilated any more. Since the phase evolution depends on the frequency, this implies that adaptation can not be reached any more.

Fixing the values of μ and ϵ is not an easy task; not knowing the external signal makes this task even more difficult. These last comments will be presented in more details, especially in the chapter about the results of the adapted frequency oscillators tempo detection; but first the introduction to the music theory.

Chapter 3 MUSIC AND ITS ANALYSIS

When listening to music, humans can tap the rhythm of the music and realize that they describe a periodic movement. Music could then be considered as a periodic signal suitable as an external input to the adaptive frequency oscillators. A periodic music signal is characterized with its frequency and its phase, when applied to adaptive frequency oscillators. As the frequency represents the tempo, the phase symbolizes the beats that are detected by human beings when tapping to the rhythm of a song. The main purpose in this project is to apply these adaptive oscillators to detect the tempo of any kind of polyphonic music with a minimum signal processing computation. Thus, this chapter introduces first the different notions linked to music and then the methods already used for tempo detection and beat tracking. The next chapter will then explain the actual implemented algorithm.

3.1 SOME DEFINITIONS

To be able to understand the musical world and then to describe a method to apply on a piece of music, some notions have first to be defined.[1]

3.1.1 MUSIC

Music can be considered as a combination of pure sine waves produced from different instruments and sounds. Each sound is characterized with a fundamental frequency, and some overtones. When producing music, this sound is composed of combination of these distinct frequencies. The fundamental frequency is then the lowest one founded. The other ones are called overtones and are either harmonics or partials. If they are integer multiples of the fundamental frequency, they are called harmonics. Otherwise, if they are integer fractions, they are called partials or subharmonics. When listening to many instruments including the human voice, signals are more or less periodic and so harmonic sounds are produced. But some of them like cymbals, tam-tams or bells are naturally oscillating at partials and so produce inharmonic tones. It is then difficult to define which of the frequency people consider as the tempo of a song.

[1]The definitions are mainly taken from the site http://en.wikipedia.org/wiki/Music_theory based on different sources like Chase, Wayne. *How Music REALLY Works!*, 2nd Ed, Vancouver, Canada, Roedy Black Publishing, 2006. or Sorce, Richard,"Music Theory for the Music Professional", Ardsley House, 1995.

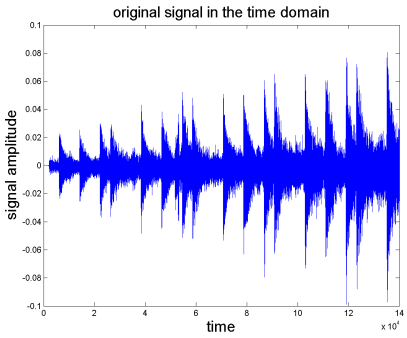


Figure 3.1: *example of an audio waveform as a function of time (System of a down, 'Bounce').*

The tempo, another musical knowledge, is the speed of a piece and is characterized by the beats among the music. The beat is a pulse that is the basic time unit of a piece. Music is then characterized by a sequence of strong and weak beats organized into meter that take part in the time signature, which is the number of beats in a measure, and gives the speed of the music that is the tempo. Are you lost? All these different notions of music are developed more specifically in the following sections. To sum up, the main purpose is actually to detect the tempo after extracting the beat of any polyphonic music; that is to find the number of beats per minute of any song.

3.1.2 RHYTHM AND METER

When talking about music, the first term to think about is the rhythm: the arrangement of sounds depending on time. It has no precise definition, but it keeps the idea of temporal regularity. A meter represents a certain amount of regular pulses, called measures or bars. The regularity of the meter defines the rhythm of a piece.

The time signature or the meter signature specifies the number of beats needed in each measure. Through increased stress and attack, strong and weak beats are created and accent can be given to particular tones. As said in [15], the "experience of rhythm involves movement, regularity, grouping and yet accentuation and differentiation".

3.1.3 PULSE AND BEAT

In [15], the pulse corresponds to the sense of "equally spaced temporal units" as said by Handel in 1989. It is defined as the basic time unit for music. When listening to the metronome, for example, each tick of it corresponds to a beat. So the beat has a very specific point in time. It represents the time when listeners will tap their foot on the ground, for example. In general, the counting of beats can enable some musicians to keep synchronized with music even if the rhythm of it is not regular.

DeLone et al. defined the pulse in 1975 as "a series of identical, yet distinct periodic short-duration stimuli perceived as points in time". It is linked to the

tempo of the piece when looking how fast the pulse is running. The beat is considered as a point that reappears regularly in time.

3.1.4 TEMPO

Tempo defines the speed of a piece. It is indicated in beats per minute (BPM). A rate of $60BPM$ means that one beat will occur every second and a tempo of $100BPM$ represents a period of $600ms$. To get a great tempo, a large number of beats have to be played in a minute and the piece is then being said to be fast. The actual number of beats per minute for a specified piece will depend on the music itself so that it is difficult to get a precise tempo. It depends on the notes used since some of them can be played more quickly in terms of beats per minute.

The tempo is not constant for a same piece of music. In fact, when playing a song, its tempo can change. It can be done gradually or immediately depending on the musicians' choice. Changing tempo implies a modification in the periodicity of the beat pulses and also on the rhythm of the song. When tapping then the rhythm of such a song, people adapt their movement to this change of tempo, as long as it behaves as a periodic function.

3.2 RELATED WORK ON MUSIC ANALYSIS

A lot of articles were written about beat tracking and tempo detection. They describe methods that have to be differentiated depending on the input signal and its causality; first, if the used data are acoustic and directly implemented from the music or symbolic and already treated, and secondly, if the method can be implemented in real-time or not. For the first differentiation, MIDI (Musical Instrument Digital Interface) is an electronic protocol which transmits only digital data like pitch or intensity of notes: it does not send an audio signal. It can also control the volume or the parameters that fixes the tempo. When using the MIDI protocol, music has already been treated and the data become symbolic.

In this project, MIDI can not be used. In fact, the data should be acoustic with a sound of any polyphonic music, and the method should be causal, so applicable in real-time. Three different methods are summarized and explained in this chapter. All of them implemented oscillators, but in a different way.

3.2.1 SCHEIRER'S METHOD

Scheirer in [15] developed an algorithm based on these two important points: causality and audio signals. He was able to find the tempo of a piece by splitting it first in different subbands which were pre-processed with filterbanks. He then analyzed them separately and convolved them to get the one with the highest energy and so finding the frequency of the tempo. Scheirer could show that a listener could not find the rhythm of a piece when either only one of the subbands is kept or when the subband envelopes are linearly dependant. In consequence, the analysis has to be done on the whole spectrum of the piece of music; no simplification can be done and in his method, the subband envelopes are kept and analyzed separately to maintain the musical rhythm.

In addition, Scheirer claimed that transcription is not needed to find the tempo of a piece. A lot of articles describe methods that segment first the music into notes, onsets and time intervals, separate then the streams of sound, and classify the timbres of the components. All these elements were then grouped, analyzed and post-processed so to get the beats, the tempo and finally the rhythm. This is a laborious work for monophonic music, so what about polyphonic pieces? Scheirer proposed in [15] a simple and basic analysis of music with a minimum of understanding of the piece. He could prove that its method worked for any type of music; actually he could show that the information contained in the amplitude envelopes of the filterbank outputs of its method was sufficient to detect the tempo. So he claimed that notes were not necessary to find the pulse, i.e. the rhythm of a piece of music.

For the onset detection, Scheirer uses 6 frequency divisions which should correspond best to human perception of sound. In fact, he tries to find the tempo which people should tap the most naturally. For each of these divisions, he applies a certain number of comb-filters. These are basically a periodic function with three impulses. The energy of these periodic functions are really high if their period is close to the one of the external signal. The comb-filter with the highest energy is then chosen and its period corresponds to the tempo of the music. The disadvantage of using these comb-filters is the fact that the possible tempo values has to be assign in advance, when fixing the period of a comb-filter. Furthermore, the number of comb-filter should be limited, because of the computation time. In fact, each subband has the same number of comb-filters and for each of them, the summation of the comb-filters energy of each frequency is computed. This limits, at the same time, the resolution of the possible tempo: it is the value of the comb-filter period that is the closest to the actual tempo of the music.

In addition, the beat has then to be found too, so to know at which moment people would tap the rhythm of the music. Scheirer uses then the phase information of the chosen comb-filter. In fact, he said that phase could predict the next impulse and this next beat is computed in its method all $25ms$. This becomes an additional computation.

To sum up, Scheirer uses a lot of memory due to its comb-filters and the precision of the detected tempo depends on the number of comb-filters implemented. His method has also parameters that should be tuned, like the frequency filterbank, the amount of comb-filters and their period, the envelope sampling rate, etc. That makes the method not very intuitive, and the different notions and parameters of the comb-filters have to be understood before using them.

3.2.2 SHIU'S METHOD

In [6], another method for on-line beat tracking is implemented and it is divided in two blocks: the onset detection and the phase-locked-loop. This algorithm detects the music tempo in real-time and finds the beat pulses. For this second purpose, a prediction is computed based on past positions of beats. In [6], the song is compared to "a quasi-periodic function whose frequency is about equal to the music tempo while the position of the individual pulse can be viewed as the phase information of a period". The phase-locked-loop is then used to track this phase, that becomes a measure of the difference between the expected beat and the beat that was performed.

The music signal is the input of the onset detection and a sequence of pulse becomes its output. The function used for the beat observation uses a filter to observe a previous and a current time window and to keep track of the beat. This enters then a phase-locked-loop system that is based on three different parts: a phase detector, a low-pass filter and a digital-controlled oscillator. The first step, the phase detector, calculates the time difference between the observed beat coming from the sequence of pulse and the estimated beat from the phase-locked-loop output. The error between these two pulses influences then an external clock that synthesizes the output signal. For example, if the error is positive, this clock must be accelerated. This allows the algorithm to accept changes in tempo, thanks to the last step, the digital-controlled oscillator. It adapts the difference computed during the loop to obtain the new estimation of the position of the beat pulse. The interval between two consecutive estimated beats gives then the period of the music piece and the corresponding music tempo.

Shiu does not explained in details how its digital-controlled oscillator works and only one song is tested in [6]. The onset detection algorithm is not presented either. This makes it hard to evaluate his method, but his results will be used for comparison, later on in chapter 4.7.

3.2.3 LARGE'S METHOD

Large developed an algorithm based on the same two points, like Scheirer and Shiu: causality and audio signals. Beat tracking was the main purpose of this model and Large tried to match as good as possible the tempo of human beat perception as Scheirer did. His method uses a network of oscillators which phase represented the beat of a music input signal and which frequency corresponded to its pulse. This phase and frequency are continuously updated, so that the beat tracking could be implemented in real-time.

In [7], Large explained its method and the main difficulties that he tried to solve, when implementing its oscillators network. The first one was rubato, defined also as the "systematic timing deviation" [7] that appears in a song and perturbs the periodicity of its signal. The second difficulty developed by Large was based on the transcription of a song, the subject that was omitted in Scheirer's method. In fact, Large tried to solve the problem, for example, when only one beat pulse misses in a song. With his method, he was able to handle these changes in tempo without influencing the general tempo detection.

Large applied oscillators to an external music signal and obtained their synchronization. In [7], it is written: "the intrinsic dynamic system of the internal oscillator are assumed to adapt to the external rhythm, accounting for the robustness of beat perception to systematic timing deviations and rhythmic complexities found in naturally produced rhythms". The external signal influences the phase and the frequency of the oscillators and changes their behavior. The oscillators were reduced to a simple dynamical system described by a phase, the momentary point of the oscillator on its limit cycle. The phase is then entrained by the external impulses signal and an expectation function is used to describe a region around $\phi = 0$. When an event falls outside of this region, the oscillator phase and frequency are not influenced. This implementation solves, for instance, the second difficulty Large described. An adjustment of the phase can be done as long as the impulse is inside the region: if $\phi < 0$, the oscillator

phase is too early and if $\phi > 0$, its phase is too late. Knowing the difference of the expected and the actual phase, adjustment is done depending on the expectation region.

Large implemented the music signal the same way as Scheirer did. In fact, as explained in [8], first he divides this signal into several frequency channels. As Scheirer uses only 6 different bands, Large uses 20 channels to get the whole frequency range of human perception. The range determines four different octaves and 24 oscillators are needed for each octave to be totally represented. Then full-wave rectification, envelope extraction and half-wave rectification are applied respectively to the samples of each channel; these are presented in chapter 4.3. Large added a last step to his onset detection: the normalization of the amplitude of the signal. This final external signal is then sent to the network of Hopf oscillators to model the different perception of beats that can appear when listen to a complex music signal. The period and phase vary for each of the oscillators, to describe the whole possible perceptible beats. It is similar to Scheirer's method when initializing the comb-filters period and it represents again a lot of computation time when using them. To find the most accurate oscillator, these dynamical systems "compete with one another" as explained in [8], to reduce the number of active oscillators. In fact, each oscillator can be deactivated by another one which has a higher energy, i.e. which is closer to the tempo of the external music signal.

Large's method uses again a lot of memory when implementing the oscillators network. Since a term adds energy to those oscillators whose activity correlates with the rhythmic input signal and with the competition, computation time can be strongly reduced. The method could then be running in real-time. Large tested it only with piano melodies, which are already very complex signals to be analyzed as he said; but unfortunately, there are no titles for a possible comparison.

3.3 METHOD USING OSCILLATORS

Using oscillators to detect tempo has some advantages: oscillators synchronization to an external unknown signal and their temporal stability to perturbations. It allows also to deal with tempo modulation. For example, small deviations from the periodic music input signal can be allowed and can be followed by an oscillator, without perturbing the detection.

Large tested its method only with piano melodies, so to simplify the external music signal to a function of discrete impulses. Moreover, its method has also some parameters to be fixed: during his tests, Large initialized the phases and the periods with the expected values. He had to know them before using its method. Another problem of Large's method is when the music signal effectively changes its tempo. In fact, with the implemented expectation function for the phase, bigger tempo changes are not detected, because they are considered as outside of the region.

Scheirer's method implies also a lot of parameters tuning. It has an additional difficulty: it has a lot of computation and finding the comb-filter with the period that is the closest to the music within all band frequencies is not an easy and obvious task. Furthermore, it is not clear if this method could really be implemented in real-time, due to its high computation time.

In all three cases, the implementations started with an onset detection: the music signal had to be transformed or sampled before it could be used by the oscillators. Large and Scheirer used the same detection, for their methods: amplitude envelopes. Shiu's method implemented a sequence of discrete pulse. Oscillators were then applied: Scheirer used a network of linear oscillators, Large a network of nonlinear oscillators and Shiu a digital-controlled oscillator. The oscillators of the two networks were implemented quite differently, but both implied a lot of memory and computational time.

A method that detects the tempo of any music signal using oscillators will be implemented. It should be insensitive to small changes in tempo, but still react to larger changes. It should also reduce the computation time, so to be used in real-time. In fact, it should be possible to play the music signal at the same time as analyzing it. This will be the objectives of the adaptive frequency oscillators method. Its different steps will be presented in the next chapter.

Chapter 4 TEMPO DETECTION

The implementation of the tempo detection using the adaptive frequency oscillator is explained in this chapter. First of all, an introduction about the music sampling is done, based on the application of the oscillators. Then adapting the frequency oscillators directly on the music input was tested. The oscillators converged, but with a really high frequency value, so that the results were not satisfying. Some explanations are given before finding another method, using pre-processing of the music input signal, when applying some of the steps of Scheirer's onset detection.

4.1 MUSIC AS AN EXTERNAL PERIODIC INPUT

As the sound is shown to be a combination of complex periodic signals and so to become a potential external periodic input for oscillators, the idea is now to find a way to use the adaptive frequency oscillators and especially their frequency adaptation capability to detect this tempo. Some explanations are given about the music sampling for the implementation. MPG123 was the chosen program: it is a real time MPEG audio player and decoder. In this case, it is used to play .mp3 files as an input for the frequency adaptation of the oscillators and as an output to listen to the music once the adaptation is computed.

4.1.1 MONO- AND STEREOPHONIC SOUND

Sounds of a monophonic signal come from a common signal path and produce a single channel of samples. It contrasts with stereophonic sounds that use several independent audio channels. The difference of path numbers comes from the song recording. In fact, when producing a two-channel stereophonic sound, two microphones record the sound source from two different places. This results then in two signals with a slightly time difference, depending on the distance between the microphone and the sound source. When listening to stereophonic signal through one channel, some interferences can be created and a loss of fidelity can appear. Thus, in this project, when using the MPG123 program with the monophonic processing option, this loss is not significant and so, stereophonic as well as monophonic sounds can be analyzed and played back after adaptation.

4.1.2 DOWNSAMPLING

In signal processing, sampling reduces a continuous signal to a discrete signal; that is, in this project, sampling is the conversion from a sound signal to a sequence of samples, that represents a value at a certain moment in time. If a continuous signal is sampled, the resulting signal measures the value of this continuous signal every T seconds:

$$x[n] = x(nT) \tag{4.1}$$

where n represents a positive integer and T the sampling period or sampling interval, which is the time between two consecutive samples. The sampling frequency is then evaluated as the inverse of the sampling period: $f_s = \frac{1}{T}$ and defines the number of samples obtained for one second.

Human hearing has a frequency range of $18-20kHz$. To respect the Nyquist-Shannon theorem, the audio signals have typically a sampling frequency of $44.1kHz$. In fact, the Nyquist theorem guarantees a perfect reconstruction of a sampled signal, if the sampling rate is more than twice the maximum frequency. In this case, the minimum sampling frequency would be $40kHz$. From another point of view, the frequency that is half the size of the sampling frequency, represents the highest frequency possible to get a perfect continuous signal. Mathematically, it is the Nyquist frequency.[1]

Since the oscillators have to adapt their frequency to one of the music, their signals have to get the same time step. It is important that all data from the music are considered, so that the timing of oscillators has to match the one of the music. Downsampling allows to reduce the data rate and so the size of the data, and thus also the time between each sample. When sampling the music in this project, the downsampling factor was fixed equal to 4, so that the sampling rate decreases from $44.1kHz$ to $11.025kHz$. The Nyquist frequency is then $5512.5Hz$. Since the rhythm is often indicated by the drum patterns that are situated in the lowest frequency bands, that is a range of $1 - 200Hz$, the tempo can largely be detected, when using this downsampling factor of 4. Goto in [5], for example, analyzed audio signals in real-time, when extracting only these drum patterns.

The time interval between two consecutive samples music becomes equivalent to the time step of the oscillators adaptation equations. In fact, Euler's method is used for the oscillators adaptation computation, and the following implementation is obtained:

$$\dot{r}(i) = (m - r(i)^2)\ r(i) \tag{4.2}$$

$$\dot{\phi}(i) = \omega(i) - \frac{\epsilon}{r(i)}\ \sin(\phi(i))\ song(i) \tag{4.3}$$

$$\dot{\omega}(i) = -\epsilon\ \sin(\phi(i))\ song(i) \tag{4.4}$$

$$r(i+1) = r(i) + \dot{r}(i)\ dt \tag{4.5}$$

$$\phi(i+1) = \phi(i) + \dot{\phi}(i)\ dt \tag{4.6}$$

$$\omega(i+1) = \omega(i) + \dot{\omega}(i)\ dt \tag{4.7}$$

[1] The explanations from this section are mainly taken from the site http://graphics.stanford.edu/ mmp/chapters/pbrt_chapter7.pdf, a chapter of M. Pharr and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, Morgan Kaufmann, July 2004.

When increasing the sampling rate, the number of adaptation computation increases too: these two notions are related.

Mathematically, the time between each step of the adaptive oscillators equations can be defined by dividing the total time of the song with its total number of data, considering its sampling rate. Or much simpler, this time is the inverse of the sampling frequency f_s:

$$dt = 1/f_s \tag{4.8}$$

It becomes equivalent to the sampling period, introduced in equation (4.1). This way, the time step ensures that each music sample is considered for each adaptation computation; the timing of the oscillators is then matched to the music data. This time step was then tested, so to avoid potential numerical errors of the program independently from the implementation.

4.1.3 THE FIVE TESTED SONGS

Five different songs were chosen randomly. They are performed by four different groups: one by "System of a down", one by "Madonna", one by "Dido" and two by "Moby". Their tempo was tested with Scheirer's method, explained in chapter 4.3, to get a first idea of the frequency value that had to be obtained with the adaptive frequency oscillators method. The values were respectively $178BPM$, $196BPM$, $206BPM$, $208BPM$ and $218BPM$ for the five songs; that means a frequency of $18.64rad/s$, $20.52rad/s$, $21.57rad/s$, $21.78rad/s$ and $24.92rad/s$ has to be reached by the adaptive frequency oscillators to get the same results.

Examples of these songs are shown in figure 4.1. The time of sampling was fixed at 100 seconds and it started at the beginning of the songs.

4.1.4 REPETITION

Repetition is needed, for example, when extracting only a few seconds of a song or simply when the adaptation asks more time to converge. This is done by reproducing the external signal a specific number of times. Unfortunately, this increases also the total time of computation and has to be used with moderation. In fact, the number of times the signal has to be repeated has to be fixed considering also the parameters of the oscillators, μ and especially ϵ. As explained in chapter 2.2.4, for example, the smaller ϵ, the longer is the adaptation, i.e. the more repetition of the song is needed.

Moreover, passing from the end of the extracted song part to its beginning can induce errors during the adaptation step. In fact, the adaptation is done continuously on the music samples and putting the end and the beginning of the song together can shift the next strong beat and so the right tempo can be lost. That is why, repetition was only used when testing the different steps of Scheirer's method off-line. As soon as the adaptive frequency oscillators method was implemented on-line, this repetition was not used any more, and the songs were played from the beginning till the end.

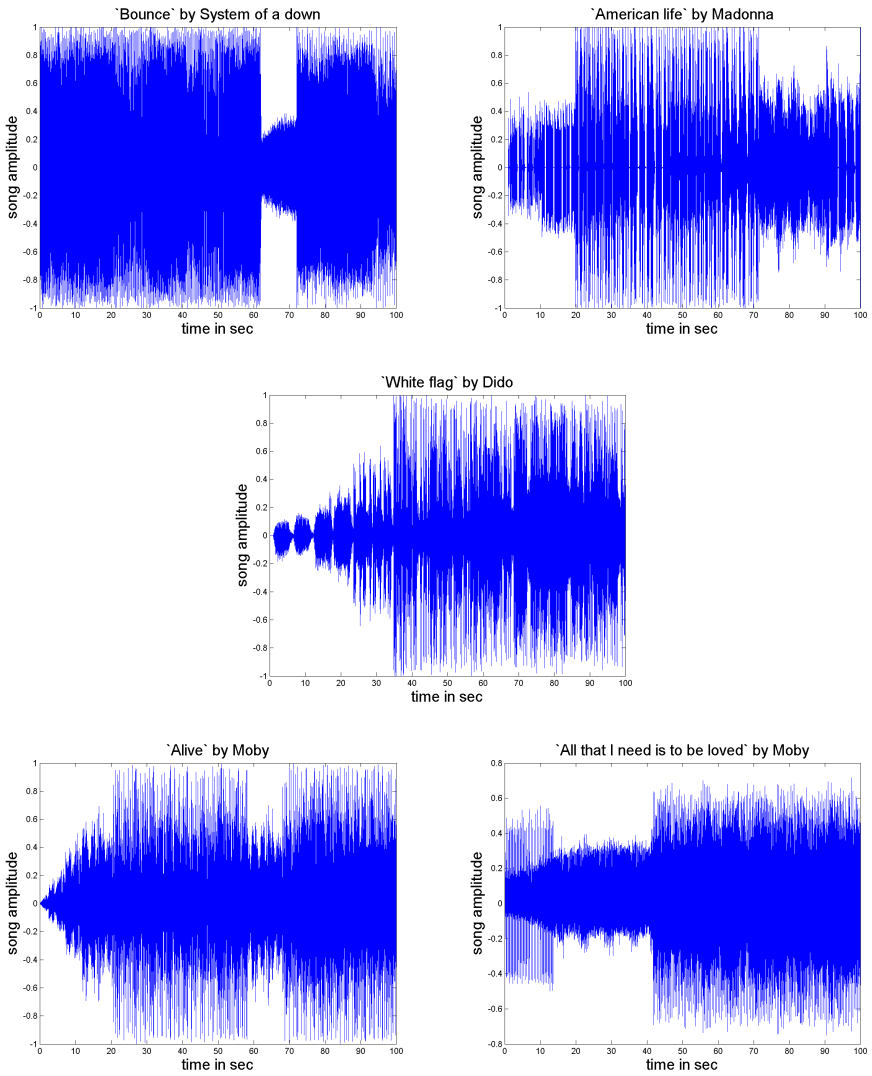


Figure 4.1: *representation of the* 100 *first seconds of the five implemented songs, in the time domain: 'Bounce' by System of a down, 'American life' by Madonna, 'White flag' by Dido, 'Alive' by Moby and 'All that I need is to be loved' by Moby.*

4.2 ADAPTATION APPLIED DIRECTLY ON MUSIC

Implementing the adaptive oscillators directly on music was the first idea. In fact, having no music pre-processing would be the best application possible in real-time: there would be no time delay between the music sampling and the oscillators adaptation. Unfortunately, the first results were a bit disappointing.

Six oscillators were chosen to be adapted to an external music signal input. Their parameters were all identical, except their initial frequency. The idea was to test if all oscillators were adapting the same frequency of the music or if there would be different possible solutions. At the same time, the values of the initial frequencies were fixed to be as different as possible, so to find the most suitable frequency.

Even if there was convergence, the results of applying the oscillators directly on music were not satisfying. In fact, the learned frequencies were much higher as the expected values. For example, for the song by "Madonna", oscillations between frequencies of 309 and $330 rad/s$ were obtained. For the one by "Dido", they were between 613 and $625 rad/s$ and for the one by "System of a down", they were between 632 and $651 rad/s$. The expected values given by Scheirer's method were respectively, $20.52 rad/s$, $21.57 rad/s$ and $18.64 rad/s$. In fact, when applying the oscillators directly on the music, all notes are considered and not only the changes in sound that represents the tempo. All notes have the same importance and so the beat and, at the same time, the tempo are not distinguished.

Applying a filter on the music samples could reduce these high frequencies. Thus, in [15], it was demonstrated that only keeping part of the music can induce error, during tempo detection. Moreover, a cutoff frequency should be fixed for a low-pass FIR filter. The signal amplitudes for the frequencies higher than this cutoff frequency are then reduced. For this project, the drum patterns could be a suitable limitation: that would fix the cutoff frequency around $4Hz$ ($240BPM$). In [5], tempo detection was done only on drum patterns, extracted from an audio signal. Goto conclude that its method could detect the beat of songs even without drums, but it needed additional musical knowledge, like beat structure or onset times. Some tests were done, at this point: for several cutoff frequencies, either the oscillators could not reach convergence, or they automatically tended to a 0 frequency. In figure 4.2, a filter with a cutoff frequency of $4Hz$ is implemented. For the song by "Dido", the two oscillators, initialized at $\omega_1 = 20 rad/sec$ and at $\omega_2 = 30 rad/sec$, for example, could not converge. The parameters were fixed at $\epsilon = 100$ and $\mu = 1$.

Changing either the oscillators or the cutoff frequencies could not give better results. Moreover, the parameters of the oscillators needed high values to start adaptation and not all of the six adaptive frequency oscillators converged systematically. That is why, the whole frequency ranges of the tested songs were kept when implementing the adaptive frequency oscillators.

To conclude, adaptation of oscillators to a music input signal needed, in this case, onset detection. Envelope extraction will be used to emphasize the change of rhythm in the music. Before that, Scheirer's method is explained: in fact, some of its steps will become a basis of the pre-processing used with the adaptive frequency oscillators method.

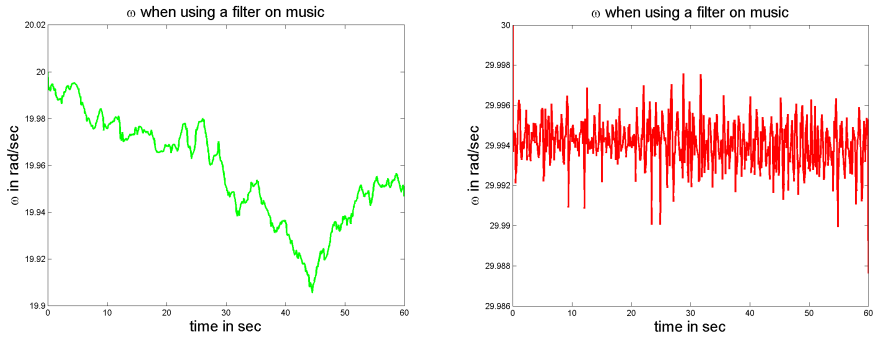


Figure 4.2: *adapted frequencies of oscillators with initial frequencies of* $\omega_1 = 20rad/sec$ *and of* $\omega_2 = 30rad/sec$*, when filtering the music with a cutoff frequency of* $4Hz$*. Adaptation could not be reached and the general results were not satisfying.*

4.3 THE STEPS OF SCHEIRER'S METHOD

Scheirer's method implemented here[2] assumes that the tempo found is associated with the entire piece of music: no tempo changes in a song can be detected. It extracts 2.2 seconds of music, the minimal amount of information to have, at least, two beats for the slowest assumed tempo, that is $60BPM$. As briefly explained in section 3.2.1, it basically emphasizes the sudden impulses of sound in the song and finds the fundamental period at which these impulses appear. The four steps describing Scheirer's method are detailed in the following sections.

4.3.1 STEP1: FILTERBANK

The first step breaks the initial music signal into six frequency bands. The six different ranges of frequency are: $0 - 200Hz$, $200 - 400Hz$, $400 - 800Hz$, $800 - 1600Hz$, $1600 - 3200Hz$, $3200Hz$ and more. This separation is done to avoid conflicts in "downbeats" of the different used instruments.

From the original signal in the time domain shown in figure 4.3, on the left, an FFT is applied to divide the music into the six frequency bands. These different bands are then defined and the inverse FFT of each of them are computed and plotted in figure 4.4, in the time domain. These six signals are then sent to the next step: the smoothing function.

4.3.2 STEP2: ENVELOPE EXTRACTOR

Extracting the envelope of the frequency bands signals allows considering only sudden changes in sound. This is possible since the main purpose is to find the tempo of the music signal. First, full-wave rectifiers are applied to each of the six bands to decrease high-frequency influence in the music; only the positive side of the envelope is used. This is done, in the time domain, by putting all

[2]The main code of this method was found on the website [http://www.owlnet.rice.edu/ elec301/Projects01/beat-sync].

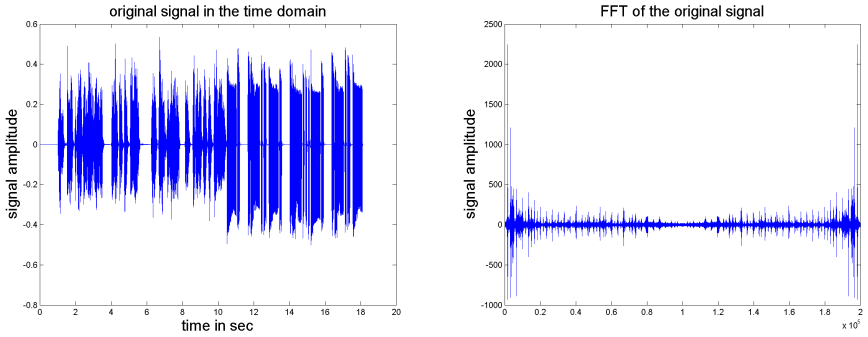


Figure 4.3: *the original music signal in the time domain (left) and in the frequency domain (right).*

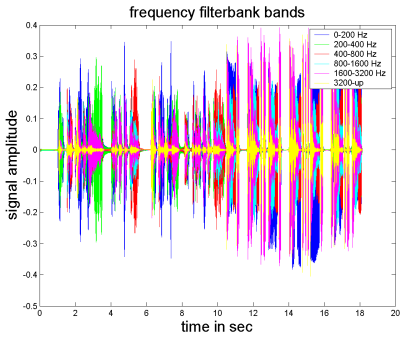


Figure 4.4: *the same music signal as in figure 4.3, in the time domain, after the separation into the six frequency bands.*

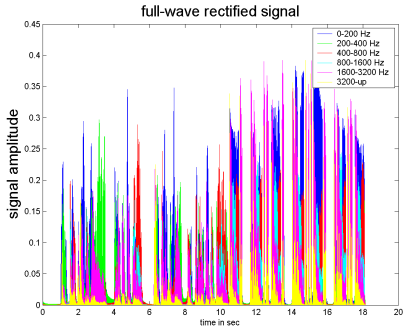


Figure 4.5: *the signal after full-wave rectification of the six frequency bands signals. After a first filtering in the time domain, only the positive side of the signal remains.*

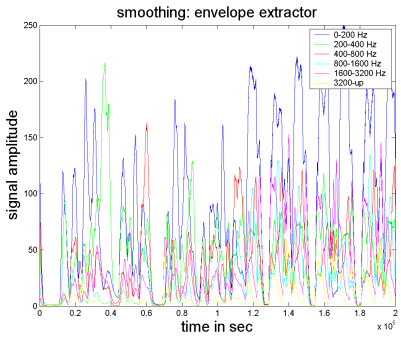


Figure 4.6: *the six frequency signals after their respective envelope extraction represented in the time domain.*

the song data of the different bands to positive values. Figure 4.5 shows the results, in the time domain, after the full-wave rectification.

Each of the six signals is then convolved with the right half of a Hanning window. It is computed with the square of a cosine:

$$f(a) = \left(\cos\left(\frac{a\ \pi}{2\ length}\right)\right)^2 \tag{4.9}$$

with *length* the total length of the Hanning window. In this case, it is fixed at $0.4sec$. The window has then a length of $200ms$; this means that the signal is smoothly brought down to zero at $200ms$. The rapid changes in slopes are kept as the slow ones are delayed and so neglected. Only the sudden changes are represented with the obtained signal that is defined as the signal envelope. This signal is shown in figure 4.6, in the time domain.

The multiplication between the full-wave rectified bands signals and the Hanning window is done in the frequency domain to reduce the operation time: a FFT is computed. The signals are then sent to the next step after being transformed back into the time domain.

4.3.3 STEP3: DIFFERENTIATOR AND HALF-WAVE RECTIFIER

Differentiating two consecutive samples of all six frequency banded signals accentuates sudden changes in sound. Only the positive differences are then retained in the time domain: this is the half-wave rectifier. The other values are equal to zero. This computation allows recognizing the emphasis of sound that represents a beat and the result is plotted in figure 4.7. The signal is finally sent to the last step to detect the tempo of the music signal.

4.3.4 STEP4: COMB-FILTERS

The signals are finally convolved with comb-filters to determine the tempo that has the highest energy. A comb-filter is a series of three impulses appearing at a defined and constant period. If this period is close to a multiple of the period of the music signal, the resulting output signal has a really high energy.

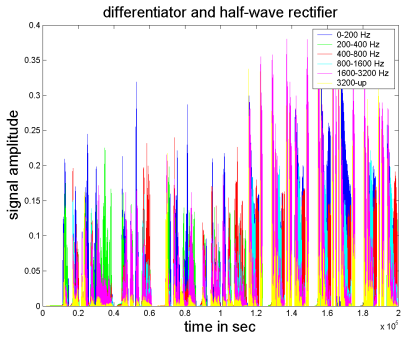


Figure 4.7: *the six frequency bands signals represented, in the time domain, before implementing the comb-filters for tempo detection.*

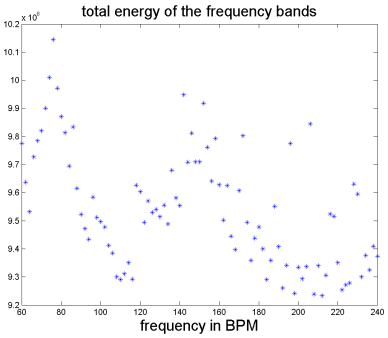


Figure 4.8: *the total energy computed when convolving the frequency bands with defined comb-filters. The fundamental tempo is found when looking for the highest value of energy given a range of expected frequencies: in this case, the tempo is* $76BPM$.

For this method, different comb-filters were defined with tempo included in a specific range of frequencies: from $60BPM$ to $240BPM$, with a resolution of 1. The convolution was computed in the frequency domain and the energy of each applied comb-filter, i.e. for each chosen tempo, is represented in figure 4.8. The maximum value is found and in this case, the fundamental tempo is $76BPM$.

4.4 ADAPTATION AFTER PRE-PROCESSING

Using Scheirer's method, different implementations with the adaptive oscillators were tested. Adding them at the end of the method, for example, was from the beginning considered as a bad solution. In fact, it would only increase the time duration of the method and not really help the computation. Since the filterbank step, separating the signal into different frequency band, was said to be necessary to avoid "downbeats" conflicts between instrument groups [15], this step was first kept. The envelope extractor needed to detect the sudden changes in sound became necessary too. As the external input of the used oscillators should have small amplitudes, the resulting signal of the envelope

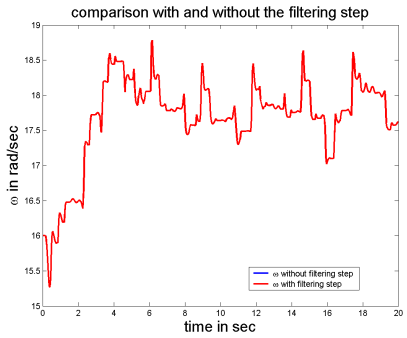


Figure 4.9: *adapted frequencies when using or not the filtering step of Scheirer's method. It can be seen that, in both cases, the adapted frequencies are exactly the same: the graphes are superposed.*

extractor was transformed to get a periodic complex signal able to be adapted. And the oscillators could be applied.

After comparing the first results of the adaptation, it can be noticed that the filtering step of the song pre-processing was actually not necessary. For the five tested songs, the adaptation was implemented with and without this steps: the results were the same in both cases. The difference of their adapted frequencies was 0. This is shown in figure 4.9, for the oscillator initialized at $\omega = 16 rad/sec$, coupled once to the original and once to the filtered song by "Dido": both graphs are superposed.

As explained in 4.3.1, this step was computed in the frequency domain and needed a FFT and its inverse for each of the 6 frequency bands. This can take some significant time and deleting this step shortens the computation time. Furthermore, in [8], it was said that low-frequency onsets had more often higher amplitudes than high-frequencies onsets. The final signal that is sent to the oscillators adaptation is reduced to a difference of amplitudes and the highest values are due to the biggest change in amplitude. So this would explain that the adaptive frequency oscillators adapt automatically to the lower frequencies, since they should have the biggest amplitudes.

The full-wave rectification, that translates all the samples to the upper part of the time axis, the envelope extraction, that is the multiplication of the Hanning window in the frequency domain, and finally the reduction of the signal amplitude, that is the differentiation of two consecutive envelope data, are the three pre-processing steps. These steps are implemented between the song sampling and the oscillators adaptation computation.

Based on Scheirer's method and first of all tested off-line, these steps are explained in more details with the song by "Dido". After the first 40 seconds, the song sampling started and lasted 30 seconds. Some more general results are then presented, with other tested songs. The duration of song sampling, in the last experiments, was also changed to compare the different obtained frequencies.

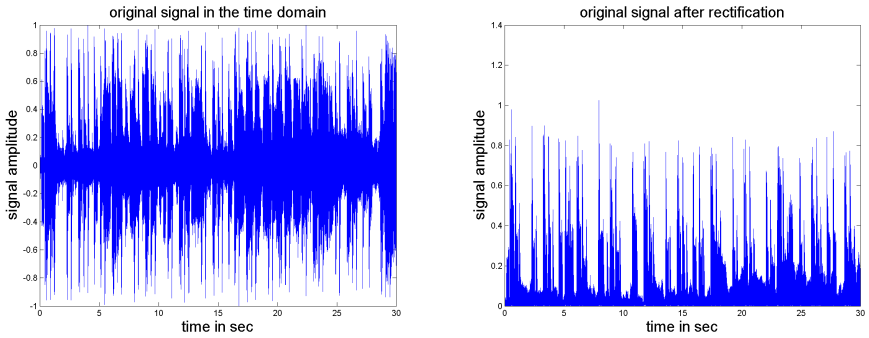


Figure 4.10: *on the left, the original input signal of the song by "Dido" is shown in the time domain and, on the right, its full-wave rectified signal. All negative values are replaced by their opposite value, to reduce the high-frequency content of the signal.*

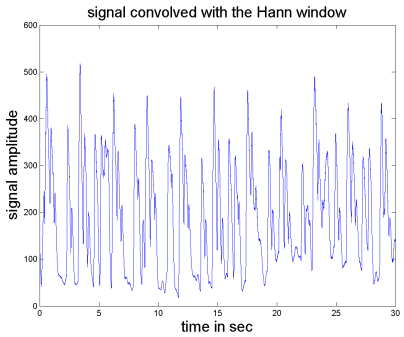


Figure 4.11: *the envelope extraction of the full-wave rectified signal. This is done when convolving this signal with the Hanning window.*

4.4.1 THE THREE PRE-PROCESSING STEPS

As the filterbank step of Scheirer's method was removed, full-wave rectification is first computed. The resulting signal is shown in figure 4.10: on the left, the original music signal and on the right, its full-wave rectified signal.

The envelope is then taken from this last signal when convolving it with the right half Hanning window of Scheirer's method; only the sudden changes are represented with the envelope. Its result is shown in figure 4.11, in the time domain.

Finally, the last pre-processing step is the differentiator of Scheirer's method, which reduces the high amplitudes of the envelope extracted signal. Retaining only the difference allows then to obtain an external input signal for the adaptive frequency oscillators. It is shown in figure 4.12: a signal with small amplitudes that still represents the significant changes in sound, i.e. the tempo to be detected.

To sum up, neither the filterbank nor the comb-filters steps were implemented for the onset detection of the adaptive frequency oscillators. Moreover, even the half-wave rectification of step 3 could be removed. The final pre-

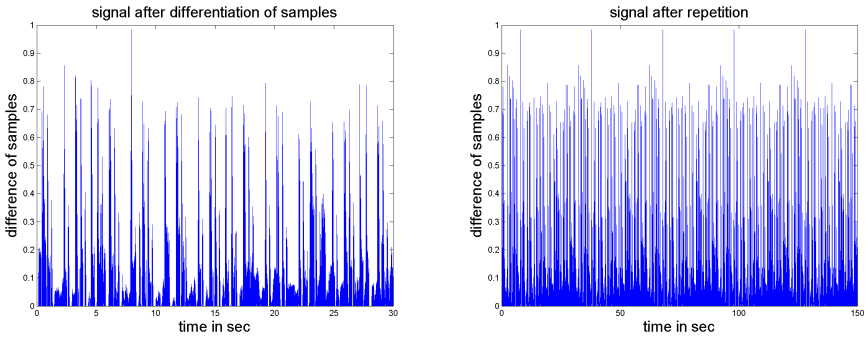


Figure 4.12: *representation of the rectified signal after the envelope extraction (left). The resulting signal emphasizes now the change in sound. On the right, this signal is then repeated* 5 *times and sent to the oscillators for the adaptation.*

processing is computed with the full-wave rectification, the envelope extraction and the differentiator. This already implies less computational time and the results are now presented.

4.4.2 FIRST RESULTS

After the third step of pre-processing, repetition and adaptation can be implemented. The results are now satisfying; they are, in fact, much closer to the results of Scheirer's method. The adapted frequencies for the different songs changed slightly when modifying the size of the song extract. This is mostly due to the warning about the repetition, warning explained in section 4.1.4 of this paper. For the song by "System of a down", the oscillators converged to $19.93 rad/s$ for 10 seconds of the song, to $19.01 rad/s$ for 5 seconds of the song and to $19.65 rad/s$ for 20 seconds. Scheirer's method gave an expected frequency of $18.64 rad/s$. When looking at the results of the first song by "Moby", the oscillators converged to $21.39 rad/s$ for 10 seconds of the song, to $21.59 rad/s$ for 5 seconds and to $21.36 rad/s$ for 20 seconds (in figure 4.13). In this case, Scheirer's method gave an expected frequency of $21.78 rad/s$.

Figure 4.13 relates the problem of the values of the adapted frequencies. In fact, it can happen that some of the oscillators converge to a ratio of the frequency of others, i.e. to a harmonic of the song. In [13], it is said that the oscillators have "rationally linked frequencies". That is the case here, for the song by "Moby". As three of the oscillators converge to $21.36 rad/s$, another oscillator converge to $10.69 rad/s$, almost half of the previous value. This can happen if ϵ is fixed to high or if the oscillators initial frequencies are to far from the expected tempo value. These issues are explained after some additional experiments, in chapter 4.6.

It has to be added that in the implemented case, the resulting frequency is only an average of the last 500 plotted data. The value is actually computed to get a first idea of the learned frequencies; it is only an approximation of the actual adapted frequency. Furthermore, the adaptation was done only for a small song extraction and repetition was used. An adaptation with the whole song should give more precise frequencies. Some general results for the adaptive fre-

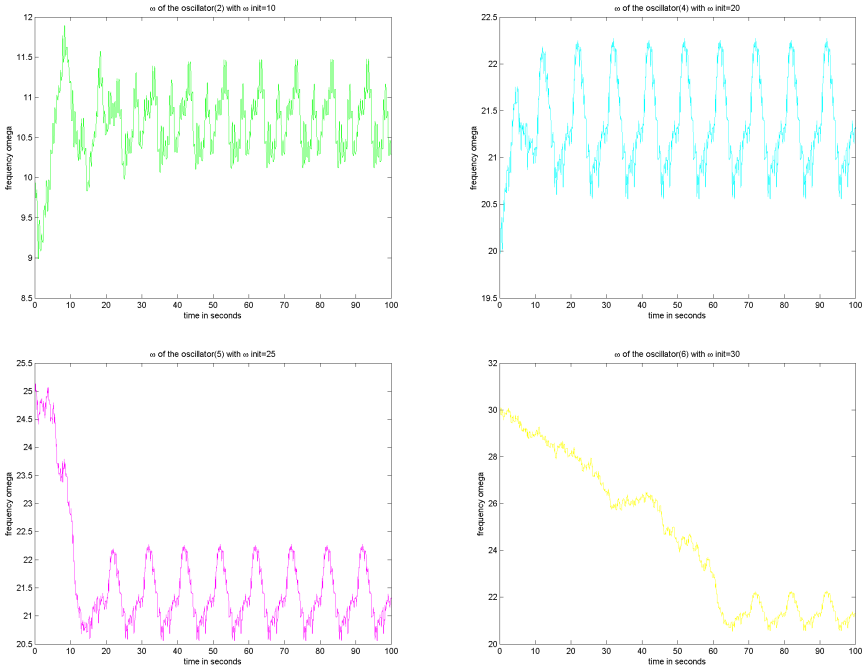


Figure 4.13: *plot of the oscillators with initial frequencies* $\omega = 10rad/s$, $\omega = 20rad/s$, $\omega = 25rad/s$ *and* $\omega = 30rad/s$. *The last three of them adapted the frequency value of* $21.36rad/s$ *as the first oscillator converge to* $10.69rad/s$, *almost half of the previous value.*

quency oscillators method are explained in the next sections, before comparing them with the results of the existing methods introduced in chapter 3.2.

4.5 ADAPTATION TIME

For these implementations, the choice of the music song can be done through the prompt, when executing the program. Every type of music could actually be tested. To be able to compare the method performances, the songs presented in section 4.1.3 were again the chosen ones for the different experiments.

All songs converged to a frequency during the sampling process. For each test, the time of adaptation and the value of the frequency were registered. In table 4.1, the initial frequencies of the three oscillators of the adaptation are changed, keeping the values of $\epsilon = 10$ and of $\mu = 0.01$. It can be noticed that the time taken till adaptation has been modified and so has also the value of the first adapted frequency. This change is still coherent with the general behavior of the frequency. In fact, it is actually predictable, since the adaptive frequency oscillators method presented in this paper returns a different frequency value each time the adaptation is computed. Looking at the adaptation time, it can be conclude: the closer the initial frequencies are from the tempo frequency, the quicker is the adaptation. The difference between the initial frequencies can also affect the time of adaptation, but never the value of adaptation. In fact, the adapted frequency, in this case the tempo of the music, is always the same value, independent of the initial conditions of the oscillators. When looking, for example, at the first adapted value for the song by "Dido", all three values are equal after the same time of adaptation.

It should also be added that for the song by "Dido", the time of adaptation is significantly higher as for the other songs. This is due to the fact that the song has a long introduction and, the instruments and the voice take a long time before appearing in the song: this has then nothing to do with the time of the oscillators convergence. That is why, this time depends not only on the oscillators initial frequencies, but also on the composition of the song itself.

The parameter μ was fixed at $\mu = 1$ instead of $\mu = 0.01$. The ratio $\frac{\epsilon}{r}$ is then modified from $\frac{\epsilon}{r} = 100$ to $\frac{\epsilon}{r} = 10$ in the second equation of the system:

$$\dot{\phi} = \omega - \frac{\epsilon}{r} F(t) \sin(\phi) \tag{4.10}$$

This changes also the time of adaptation. Depending on the song, this time is either reduced as for the second song by "Moby" (from $8sec$ to $6sec$) or arisen as for the first song by "Moby" (from $17sec$ to $28sec$) or for the song by "Madonna" (from $8sec$ to $23sec$). What is more, the oscillators converge to a value for each song when keeping the same initial frequencies, that is $\omega_1 = 7rad/s$, $\omega_2 = 9rad/s$ and $\omega_3 = 11rad/s$; except for the one by "System of a down". There was actually no convergence at all, for this song. The song was not long enough for the oscillators to reach a common adapted frequency. When increasing then the initial frequencies of the oscillators, adaptation could be reached again. This is due to the fact that this song has a higher detected tempo: as all other songs converge to a tempo around $100BPM$, this song has a detected tempo around $190BPM$. The closer the initial frequencies are from the tempo, the quicker is the adaptation. In table 4.2, some different adaptation times can be observed.

ADAPTATION TIME	**Oscillators freq(rad/s): 7, 11, 13**	**Oscillators freq(rad/s): 5, 15, 25**	**Oscillators freq(rad/s): 18, 21, 24**
System of a down	8 sec	20 sec	7 sec
Madonna	8 sec	24 sec	24 sec
Dido	41 sec	41 sec	41 sec
Moby (alive)	17 sec	28 sec	26 sec
Moby (love)	8 sec	10 sec	10 sec
1st ADAPTED FREQ			
System of a down	168 BPM	193 BPM	190 BPM
Madonna	97 BPM	104 BPM	104 BPM
Dido	86 BPM	86 BPM	86 BPM
Moby (alive)	74 BPM	143 BPM	142 BPM
Moby (love)	135 BPM	137 BPM	137 BPM

Table 4.1: *the first table shows the different time taken by the oscillators to reach adaptation: it is the moment when all oscillators have the same adapted frequency. The second table shows the frequency first value in BPM, when the adaptation is reached.*

ADAPTATION TIME	**Oscillators freq(rad/s): 9, 11, 13**	**Oscillators freq(rad/s): 13, 15, 17**	**Oscillators freq(rad/s): 17, 19, 21**
System of a down	no adaptation	37 sec	10 sec

Table 4.2: *this table shows the changes in adaptation time for the song by "System of a down" when modifying* $\mu = 0.01$ *to* $\mu = 1$*. It can be noticed that, for the first example of initial frequencies, the song was not long enough for the oscillators to adapt.*

Unfortunately, all these results imply that knowing the expected tempo of the song makes the adaptive frequency oscillators method more efficient. This has also another conclusion: the performance of tempo detection depends not only on the song and its length, but also on the parameters and the initial conditions of the adaptive frequency oscillators.

4.6 ADAPTATION RESULTS

When looking at figure 4.14, the difference of adapted frequencies can be observed, when changing the parameter μ. The adapted frequencies become more stable and changes in tempo are done more smoothly. The oscillators are less sensitive to a small change in tempo. This is presented the most significantly with the first song by "Moby". As the tempo varies a lot when $\mu = 0.01$, it became almost constant with $\mu = 1$. When looking at the second song by "Moby", it can be noticed that a change of μ does not influence the value of the adapted frequency. In fact, both tempos are equal. When increasing the value of μ, so to reduce the effect of the external signal on the phase derivative (equation 4.10) of the system, it implies more precision. It assure a smoother frequency adaptation. In fact, search of the tempo is done more systematically, with smaller steps.

Two different inputs were then coupled to the adaptive frequency oscillators to test the value of the adapted frequencies. The first one was composed with two different impulses which had both a $1Hz$ frequency, but with a time delay of 0.75 seconds between them. This input was used to understand how the adaptive frequency oscillators are influenced when an irregular rhythm drives them. The second input was composed with an impulse function whose frequency changed all 30 seconds: first a $1Hz$ frequency, then a $2Hz$ frequency, and finally a $1.5Hz$ frequency. This input was used to see how the adaptive frequency oscillators behave with a change in tempo.

When testing the method with the first input, the results were as hoped. The oscillators adapted a tempo around $1Hz$. This was actually the only periodic signal possible for this input. The oscillators were not influenced by both impulses at the same time; only one of the impulses drove them. This means that, if a second rhythm appears in the middle of a song, delayed with the tempo of the song, the adaptive frequency oscillators should not be perturbed. They should only adapt to periodic signals.

When testing the method with the second input, the results showed that the adaptive oscillators could follow a change in tempo. In fact, after a transient state, the oscillators converge to the changed tempo again. The adaptation time depends on the parameters of the oscillators, as explained in the previous section. Different experiments were done changing the value of ϵ. The value of the adapted frequency became closer to the music tempo and the adaptation time was shortened, when the value of the ratio $\frac{\epsilon}{r}$ was increased. However it became also more variable. In figure 4.15, the second input function was used as the external input signal of the adaptive frequency oscillators. ϵ was initialized at $\epsilon = 5$, $\epsilon = 10$, $\epsilon = 20$, $\epsilon = 50$ and finally at $\epsilon = 100$.

The adaptive frequency oscillators method can be implemented with external inputs that are composed either with irregular tempos or with changing tempos. In both cases, the oscillators converge to the expected tempo, depending on

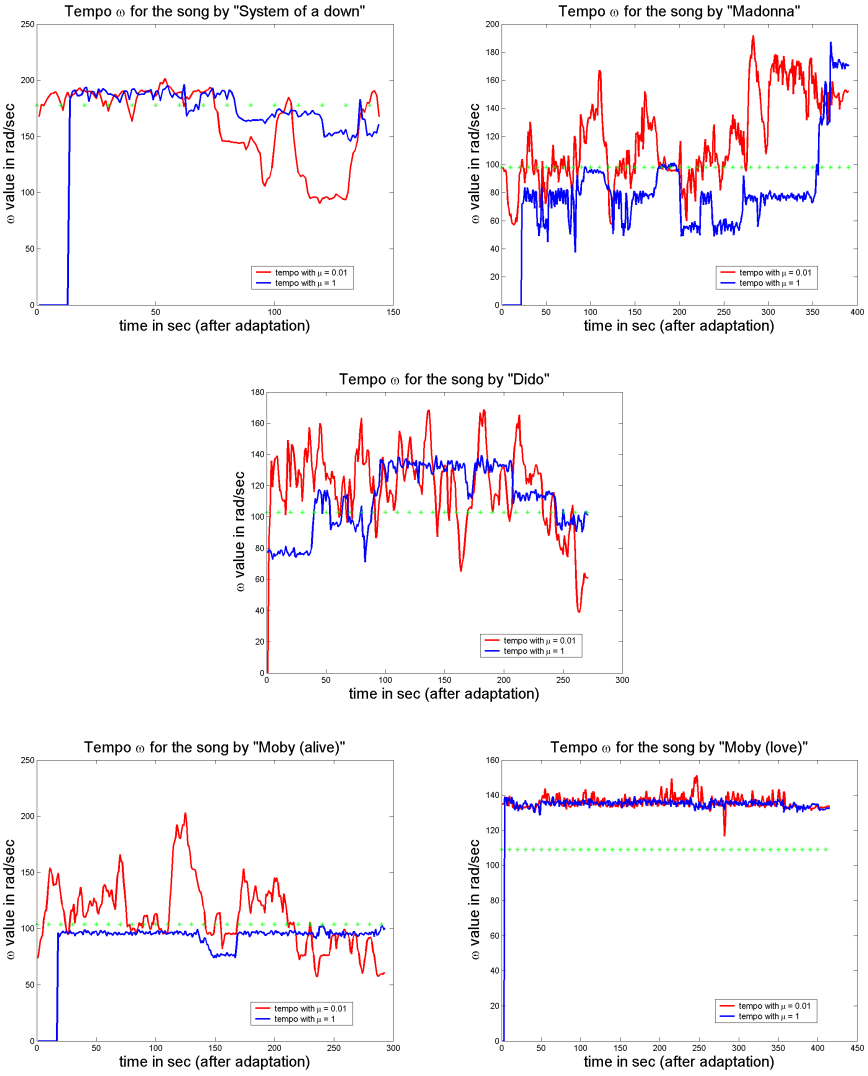


Figure 4.14: *the adapted frequencies obtained during the adaptation for the five songs, in the time domain: 'Bounce' by System of a down, 'American life' by Madonna, 'White flag' by Dido, 'Alive' by Moby and 'All that I need is to be loved' by Moby. Each plot shows the difference of tempo when changing the parameter* $\mu = 0.01$ *to* $\mu = 1$.

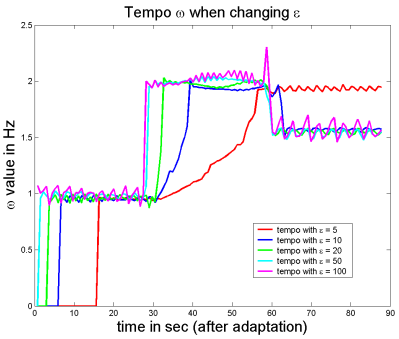


Figure 4.15: *the different adapted frequencies of an input that changes frequency all* 30 *seconds, when changing* ϵ*. As this parameter is increased, the adaptation is reached more quickly and the value of the adapted frequency becomes closer to the input frequency, but it is also more variable.*

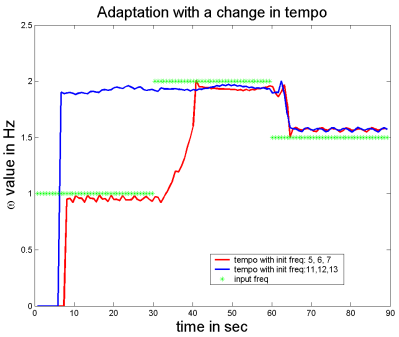


Figure 4.16: *adaptation of the oscillators frequency to a changing tempo of an impulse function, when different initial frequencies are implemented. The tempo starts at* $1Hz$*, then accelerates to* $2Hz$ *after* 30 *seconds and finally decelerates to* $1.5Hz$ *after* 30 *seconds.*

their parameters, especially μ and ϵ, and their initial conditions. To show this last point, another experiment was done with the second input. The initial oscillators frequencies were fixed first at $\omega_1 = 5rad/s$, $\omega_2 = 6rad/s$ and $\omega_3 = 7rad/s$, close to $1Hz$, and then at $\omega_1 = 11rad/s$, $\omega_2 = 12rad/s$ and $\omega_3 = 13rad/s$ close to $2Hz$. The adapted frequency, in the first case, oscillated around $1Hz$, as the frequency of the input. In the second case, it oscillated around $2Hz$: that is twice the tempo of the input. It adapted then a partial or a subharmonic, a integer multiple of the music tempo, as explained in chapter 3.1. This is shown in figure 4.16.

Again the same conclusion as before can be given: fixing the parameters of the adaptive frequency oscillators as close as possible to the expected tempo makes the adaptive frequency oscillators method more efficient and more precise. In addition, the adapted frequency can represent an integer fraction or an integer multiple of the effective music tempo. It is then not an easy task to define which tempo should be considered as the one of the input.

4.7 EXISTING METHODS

A comparison of results is then done, between the adaptive frequency oscillators method and the other ones presented in the third chapter of this paper. As the implemented algorithm of Scheirer's method could be found, comparison with his method was possible, but with some restrictions. In [6], a song was also analyzed and its results explained; so a second comparison was possible. Unfortunately, for Large's method, there were no concrete results to be compared with. In fact, Large used piano melodies, implemented as impulse functions that he considered as the input music signals for his oscillators. He added then variation on these melodies, but no results on music songs were presented.

4.7.1 COMPARED TO SCHEIRER'S METHOD

When comparing Scheirer's results and the one obtained with the adaptive frequency oscillators, there was relatively good matching. For the song by "System of a down", Scheirer's value is the original value; for all other songs, this value was divided by 2. This means that Scheirer's method returned a faster tempo than the adaptive frequency oscillators method, a tempo that is too quick for a human being to be considered as a natural beat perception. This problem was also relevant in Large's method [8]. The results are plotted in figure 5.1 where the stars represent the tempo of Scheirer's method.

Only one song has a significant different tempo value as the one indicated by Scheirer's method: the second song by "Moby". Curiously, it is the song that has the most stable tempo detected by the adaptive frequency oscillators method. The difference between both tempos is about $25BPM$ ($0.4Hz$). For the first song by "Moby" and the song by "System of a down", the results are closer. Once Scheirer's method gave a higher value and once a smaller one, respectively. For the song by "Dido", the comparison is more difficult. In fact, the adaptive frequency oscillators method shows tempo changes that can not be detected by Scheirer's method implemented for this paper. Further experiments should be done, to show the performance of the adaptive frequency oscillators method.

4.7.2 COMPARED TO SHIU'S METHOD

When comparing the results of the only song that Shiu tested in his paper [6], it can be noticed that it is close too. In fact, Shiu applied a song by "Oasis" (*Don't look back in anger*) to its beat tracking method, for the first $60sec$ of the song. He found a mean period of $728msec$ that corresponds to a tempo varying around $82.4BPM$. Shiu calculated a mean period because the music period changes over time. The same song was then tested with the adaptive frequency oscillators method and the result is plotted in figure 4.17, when dividing the tempo by two. In fact, the obtained tempo was this time twice as big as the tempo found by Shiu. The frequencies are registered all seconds for the first 60 seconds. Adaptation lasted in this case $20sec$.

Shiu found tempo values between $700msec$ and $750msec$. This corresponds to $79.2BPM$ and $84.9BPM$. It can be noticed that with the method use in this paper, the tempo is pretty close to Shiu's one, when divided by two, as said before; it is around $85BPM$. But only comparing one song is not significant for

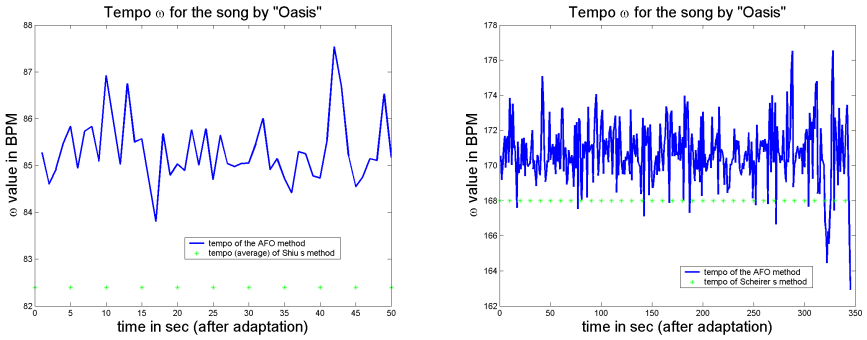


Figure 4.17: *comparison of results of the adaptive frequency oscillators method, with both the results of Shiu's method (left) and of Scheirer's method (right). For the left one, only the* 60 *first seconds of the song was implemented as in Shiu's test. For the right plot, the whole song is shown with the original tempo values.*

a good comparison. Unfortunately, no other song was used in [6] and so, this result can not be valued correctly.

Moreover, in figure 4.17, on the right, the original values found with the adaptive frequency oscillators are plotted. The value returned by Scheirer's method is added to it and compared: once more, both tempos are pretty close. As the first ones are around $170BPM$, Scheirer's method returned a value of $168BPM$ for the same first $60sec$. This was the best result of all experiments with songs.

Chapter 5 SIMULATION

Knowing that a method using adaptive frequency oscillators could give comparable results to the existing methods, the implementation of an adaptation in real-time is considered and explained in this chapter. To see what the adapted frequencies really represented, some simulations were performed. The drumming task of the humanoid robot is explained. The difficulty of matching the movement of the robot arms with the beat pulses becomes relevant. What is more, the adapted frequency should be limited to $2Hz$ ($120BPM$) to be described as naturally as possible, since a range of 1 to $2Hz$ represents the most likely frequencies people perceive as the rhythm of music [8]. This will lead to the implementation of the frequency limitation.

5.1 ADAPTATION IN REAL-TIME

The advantage of the adaptive frequency oscillators is their continuous adaptation. As long as there is an external signal applied to them, they adapt their frequency. One of the purposes to do the detection on-line is the change in tempo that can happen in a song. In fact, the rhythm of a song can accelerate or decelerate and so does the tempo. If the signal is modified, the oscillators adapt their frequencies to the new external input changes. As the oscillators in Large's method [15] can deal with small tempo deviations, they can not synchronize to large tempo changes. With the adaptive frequency oscillators method, it should still be possible for the oscillators to detect these tempo changes.

The challenge of the on-line adaptation applied in real-time is the fact that music, its onset detection and the movements of the robot arms are all functions that have to run simultaneously and continuously. The music is then sampled with MPG123 as explained before, in section 4.1: it can be implemented in real-time. The tempo detection and its adaptation are then done as quick as possible to keep this real-time. The last function, the movement of the robot arms, is then added too. To run these functions all simultaneously, parallel computing is introduced and threads are implemented. These notions are first explained, before introducing the implementation of the drumming task.

5.1.1 PARALLEL COMPUTING

There are different models available to implement parallel programming and a combination of some of them is often used. There is for example shared memory, threads or message passing.

The main idea of parallel computing is to divide a whole program in different parts that can be solved simultaneously using separated resources. Each

part has its own instructions that are executed separately from the other parts, so that different computations and operations are done at the same time independently. Since they have to solve a common problem, communication is then needed to exchange data between these different parts. Synchronization becomes an important point that has to be controlled. In fact, when communication is required, synchronization between tasks can be necessary to avoid loose of information. It coordinates the different parts of the whole program, so that the general behavior of it is respected. The disadvantage of synchronization is the resulting time spent by the parts in waiting for communication instead of completing their instructions. In fact, it is often implemented by fixing a point where a part is blocked till another part has reached a specific defined instruction or variable. This involves then waiting and increases the execution time of the different parts of the whole program.

Furthermore, synchronization has to be explicitly defined by the programmer, so that the general implementation is totally controlled. When one of the parts wants to access a data from another part, the programmer is responsible of the correct behavior and the result of the access.[10]

5.1.2 THREADS

The threads are one of the parallel computing model that was chosen in this project. The main program creates and starts multiple and concurrent paths, similar to subroutines, and called threads. These are procedures that run independently from the main program and also from each other. The main is responsible of loading all required resources for the threads. It remains as long as threads are used, so to be sure that the shared resources are accessible. Each thread has then local data, local instructions and local operations. It shares also all the resources of the main program and benefits from a global memory view since it is initialized in the main program, as all of the threads. They can then all access the same global variables, and this access is controlled through synchronization when communication is needed. Communication is, in fact, done through global memory: the threads update the address location of a global variable. Synchronization, as explained before, insures that only one thread can modify this global variable at any time. To block and consequently to protect the access to global data, semaphores or locks are used. The first thread that acquires the lock can then safely read a data or write into it. When another thread ask to lock the same data, it has to wait till the data is unlocked by the thread that owns its semaphore. This creates time delay that can become significantly high. The modifications done by one thread are then visible to all other threads as long as the changes are done on the shared memory. It is important to keep the number of threads as small as possible, because difficulty in designing shared memory processes increases with the number of threads[11].

"POSIX Threads" or "Pthreads" is the library used in this project. The idea of shared memory is implemented and the semaphore is in this case managed with "mutex", abbreviation for "mutual exclusion". It deals with synchronization and has the effect of protecting access to a shared data resource, that is, in general, a global variable initialized in the main program. Only one thread can lock it at any time and so the security of data is ensured. Since a thread has to wait to lock a mutex, synchronization can be implement by controlling

thread access to data.[1] For example, in this project, synchronization has to be ensured for the music samples: so the music sampling thread has to wait till the adaptation thread has read the samples to make sure that all the data are considered. The whole implementation is explained in more details in the next section.

5.1.3 IMPLEMENTATION

The main program initializes the threads. First of all, three threads were implemented: the music sampling, the frequency adaptation and the robot arm movements. The music thread begins with the song, as the adaptation starts after 3 seconds, so that the whole table of the external signal samples fixed at almost 3 seconds is totally initialized. In fact, when considering the frequency of $1Hz$ or $60BPM$ as the slowest tempo possible, based on human beat perception, a minimum of 2 seconds is needed to be sure to have 2 beats during this interval of time. Thus, at least, one whole tempo period is observed. Furthermore, only a small amount of music samples should be sent to the oscillators. This allows to listen to the music at the same time as adaptation is computed. Since this amount of new added samples is less than the "3 seconds" table sent to the oscillators, each new sample has to be added at the end of this table, so that the song is respected. In this implementation, the size of the song data given to the oscillators is 32678 samples per seconds and the length of data taken from the song at each step is 2048 samples per seconds.

The last thread, that is the implementation of the robot arm movements explained in the following section, is started when all adaptive frequency oscillators have found the same frequency value. The number of oscillators was reduced to 3, since the tests done in real-time shown that all six oscillators always converged to the same frequency. Their initial frequencies were then fixed at $\omega_1 = 7rad/s$, $\omega_2 = 9rad/s$ and $\omega_3 = 11rad/s$. These correspond to a tempo around $1.5Hz$ or $90BPM$ and describes the middle of the drum patterns, that are actually in a range of $60BPM$ to $185BPM$, as said in [5]. These patterns generally represent a good basis of the song tempo. It has to be added that the adapted frequency should not be greater than $2Hz$ due to the limitations of the robot; this corresponds to a value of $120BPM$. If this happens, the frequency sent to the robot has to be reduced. This is explained in the following section.

All three threads were then running simultaneously till the end of the song. In fact, when the song ends, the music thread stops the two other threads and the main program closes. The communication between the threads are done trough two mutexes: the music mutex between the music and the adaptation threads, and the frequency mutex between the adaptation and the robot threads. The second one is asynchronous, so that their is no waiting between the two threads; the adaptation thread saves the obtained frequency value in the global variable which is read by the robot thread only when needed. In contrary, for the music mutex, synchronization is needed: all music samples should be safely transmitted to the adaptation thread. This transfer creates a time delay and makes it difficult to have a real-time music sampling. Moreover after some tests on the time taken by both programs, the sampling and the adaptation, it could

[1] The explanations from this section are mainly taken from the website https://computing.llnl.gov/tutorials/pthreads/

THREAD 1	**MUSIC-ADAPTATION**
Read 2048 song samples, saved at the end of the 32678 data	
After 32678 samples, adaptation	- high frequency content reduction - envelope extraction - amplitude reduction - oscillators adaptation - test oscillators frequency
If oscillators have the same frequency	start the robot thread (done once) save the adapted omega
Play the 2048 samples to hear the song	
Repeat till the end of song	
If the end is reached	stop the robot thread
MUTEX	communication between the music/ adaptation and the robot threads
	global variable: "adapted omega"
THREAD 2	**ROBOT MOVEMENT**
Move the robot arms Stop when reaching the end of song	read adapted omega when needed

Table 5.1: *schema of the implementation, showing the two different threads running in parallel and the mutex that is needed for the communication between them.*

be noticed that it was pretty close. In fact, the adaptation took almost the same time as the music sampling. This is also shown when sending the music samples to the sound output when adapting the tempo: the song is playing without perturbation or delay.

These two threads were then put together; only one mutex was left, the one between the adaptation and the robot arm movement. In fact, the synchronization for the communication between the music and the adaptation was not needed any more. The music sampling and the frequency adaptation are done in the same thread and there is no delay any more. To sum up, table 5.1 shows the different instructions done by each thread and how they communicate together.

5.2 THE DRUMMING TASK

S. Degallier implemented a controller in [4] for a humanoid robot, so that it plays drums at a specified tempo. This controller uses nonlinear dynamical systems to compute trajectories of the robot arms in real-time.

As explained in [4], two different systems are implemented simultaneously: "a discrete system with a single point attractor which generates discrete trajec-

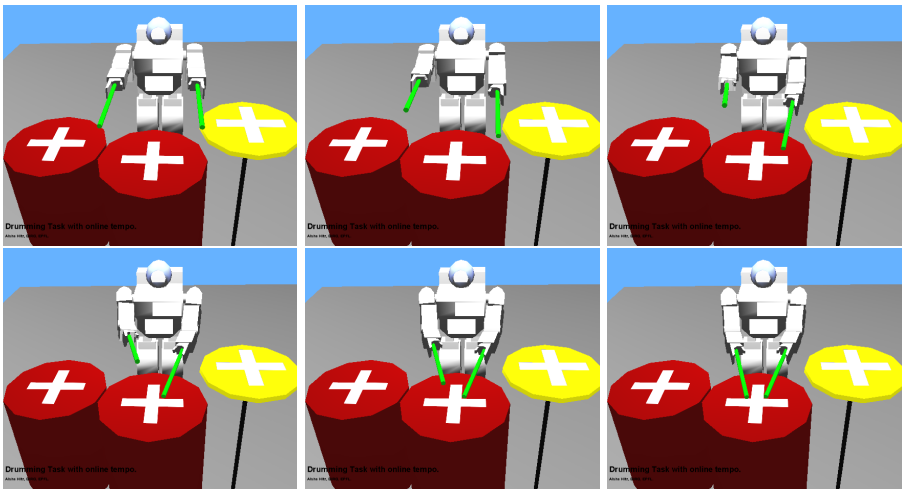


Figure 5.1: *the discrete system of the drumming controller generates discrete trajectories computed to move the robot arms to the correct instrument.*

tories towards a goal and a rhythmic system which generates periodic movements of controlled amplitude and frequency around the discrete trajectories". The arms trajectories are then divided in two different motion: the first one moves the robot arm upon the instrument to be hit and the second one produces an up-and-down movement of the arm to hit the instrument. The discrete system is then needed at the beginning of the drumming task and each time the robot should hit another instrument. In this project, only the middle drum is used, so that its generated trajectories are only used once, when starting the robot arms movement. This is shown in figure 5.1, when implementing the drumming controller in Webots.

In contrary, the rhythmic system is the most important one and is considered now. It implements a Hopf oscillator, as described in chapter 2.2.1, with the equations in Cartesian coordinates:

$$\dot{x} = \alpha\left(\mu - \left(x^2 + y^2\right)\right) x - \omega y \tag{5.1}$$

$$\dot{y} = \alpha\left(\mu - \left(x^2 + y^2\right)\right) y + \omega x \tag{5.2}$$

where μ controls the amplitude of the oscillations and so the movement of the robot arms, α influences the time taken to reach the limit cycle and ω represents the frequency of the Hopf oscillator and so the tempo of the music. More details about the generation of the trajectories can be found in [4].

Four degrees of freedom for each arm are used for the drumming task: three degrees for the shoulder and one degree for the elbow. The rhythmic part influences only one of the shoulders degree. It produces the up-and-down movement of the arm, so to recreate the beat upon the instrument. This is explained in the next section.

5.2.1 HITTING THE DRUM

The importance of phase locking, and especially of zero phase difference, becomes relevant at this point. In fact, a periodic music signal is characterized with its frequency and its phase, when applied to adaptive frequency oscillators, as said in chapter 4.1. The phase represents the beat pulses and becomes a function in time. Thanks to its periodicity, it can estimate when the next beat will occur in future. Moreover, when the adaptive frequency oscillators is adapted to the external input signal, their phase difference oscillates automatically around 0. These oscillators have then the same phase as the music signal. No additional computations are needed to detect the position of "downbeats", as it is the case for Large's and Scheirer's methods.

Since the arms movements are implemented with oscillators, these have also to get zero phase difference with the adaptive frequency oscillators, i.e the music signal. At the beginning of the drumming task and each time the intrinsic frequency of the arms oscillators is changed, the phase delay between the oscillators of the drumming task and the ones of the tempo detection must be equal to 0. Knowing the phase evolution of the adapted frequency oscillators and saving the value of the momentary phase position at the tempo change, allow synchronization between the two types of oscillators. When the zero phase difference is reached, the robot hits the drum when the "downbeats" appear.

Keeping the arms at a fixed position till zero phase difference is reached between the arms and the adaptive frequency oscillators allows a control on the motors and their encoder. In fact, this avoids arms movements that could imply sudden acceleration in the motors. It is actually the only moment when the robot arms could be unpredictable. When a change in tempo happens, during an up-and-down movement of an arm, this movement is finished keeping the intrinsic frequency of the arms oscillators. The drumming task is stop till the phases of both oscillators, the adaptive frequency and the arms oscillators, are equivalent. Knowing the arms positions, i.e. the arms oscillators phase, and the phase at the tempo change, the time when starting the next up-and-down movement can be computed. The robot can then accomplish its drumming task again and it should hit the drum at every beat pulse, with the new tempo value. So, controlling[2] the change in tempo makes the arms movements and thus, their motors behavior, safe.

The partition to be played is transformed "into time-varying parameters controlling the dynamical systems", as said in [4]. In this project, the partition switches the robot arms for each beat pulse: once the left arm hits the middle drum, once the right arm does. This is shown in figure 5.2. The reading of the partition is also dependent on the music tempo. It should be modified, at the same time as the intrinsic frequency of the Hopf oscillator. A change in tempo influences then both the reading of the partition and the frequency of the oscillations. Thus, small changes should be controlled, so to avoid unnecessary perturbations. (For more details about the reading of the partition, i.e. its parametrization, see [4].)

When using a function based on discrete impulses happening at a constant frequency of $1Hz$ as the external input signal, the movement of the robot arms can be plotted. As said before, the arms are implemented to be used one after the other. Before hitting the drum, the arm lifts in the air; this is shown in

[2]This last part could unfortunately not be implemented, for lack of time.

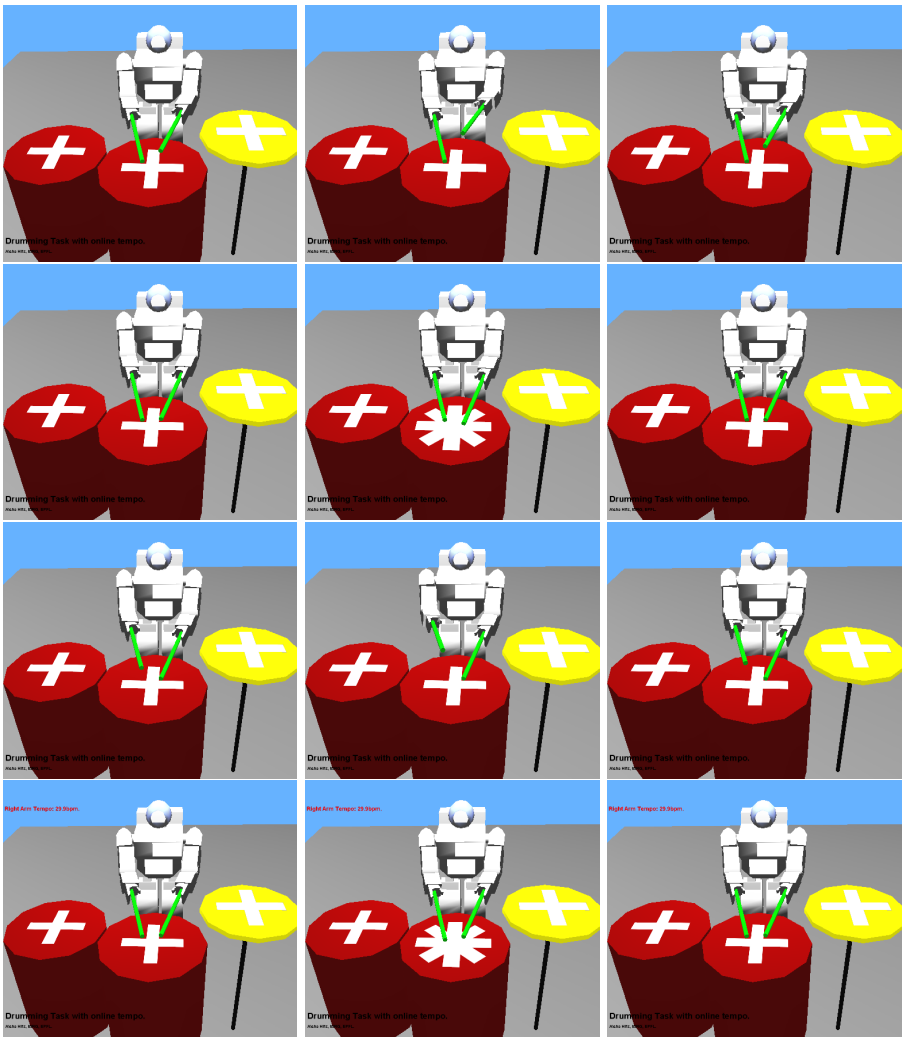


Figure 5.2: *drumming task, done by the humanoid robot, when implemented in Webots. It switches its arms for each beat pulse: once the left arm hits the middle drum, once the right arm does.*

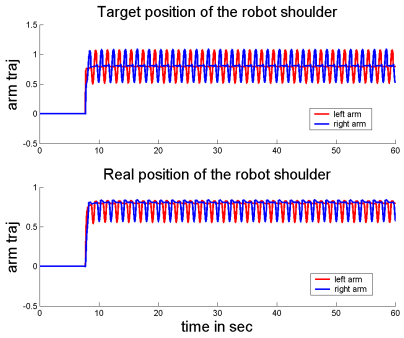


Figure 5.3: *behavior of the oscillators of both robot shoulders, when a* $1Hz$ *frequency tempo is applied for tempo detection. A switch between the movements of the left and the right arms is then shown and the hitting of the drum is represented in the second plot, with the truncated picks.*

figure 5.3, when a pick appears with values that decreases first. When comparing both the target and the real trajectories, it can be noticed that the pick with values that increases first happens only in the first trajectories. In fact, it is represented in the second one by a truncated pick, i.e. a horizontal line: that is the moment when the robot hits the drum. A longer horizontal line is then observed. It represents the moment when the arm is stationary. Since figure 5.3 shows both arms trajectories, it can be noticed that as the left arm hits the drum, the right arm is, in fact, stationary.

5.2.2 FREQUENCY LIMITATION

People have a preferred range of frequencies for the tempo. In general, they tap the rhythm of a song for values between $1Hz$ and $2Hz$. When implementing the tempo detection to move the robot, the frequency sent to compute the trajectories of its arms is limited at $2Hz$: first, because of the limitations of the motors and their encoders, but also to make the movements of the robot more human-like. The implementation when limiting frequencies higher than $2Hz$ is explained.

When the adapted frequency approaches the $2Hz$, it could have two different behaviors: either the frequencies were already oscillating around $2Hz$ and so the frequency only occasionally exceeds $2Hz$, or the frequency increases to exceed the limitation and so, the frequency has to be controlled. In the first case, the previous frequency, that was less than $2Hz$ is sent to the robot. If the adapted frequency is then still higher than $2Hz$, it is considered as the second behavior. The sent frequency is then divided by two; that means it will oscillate around $1Hz$. Since there is a big gap between $1Hz$ and $2Hz$, this transition is done in three steps. First, the previous value is sent to the robot. Then, an intermediate frequency is sent to the robot, and finally, half of the actual value of the frequency is given to its oscillators. The first step is needed to be sure that the frequency is going to stay higher than $2Hz$. The other two steps create a smooth transition of the frequencies to be adapted by the arms oscillators.

Since the oscillators adapt their frequency to the tempo of the music, dividing

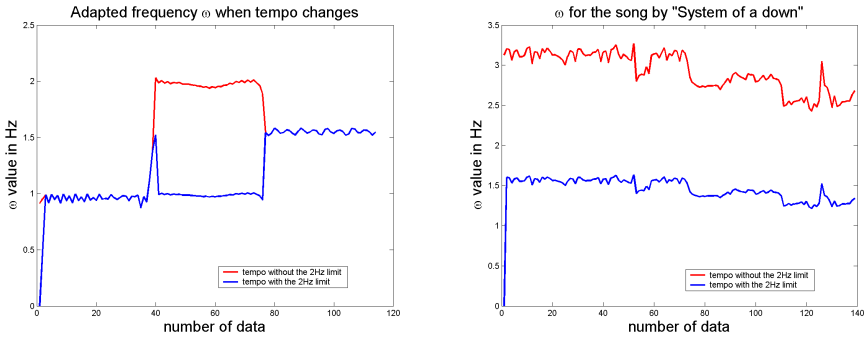


Figure 5.4: *frequency limitation for the song by "System of a down" (right) and a function where the tempo changes from* $1Hz$ *to* $2Hz$ *and from* $2Hz$ *to* $1.5Hz$ *all* 30 *seconds (left). All frequencies higher than* $2Hz$ *are reduced to half of their values.*

their adapted frequency by two would only create a perturbation and adaptation would be lost. In fact, the oscillators would adapted again this tempo of $2Hz$, since it has stronger beat pulses as the tempo of $1Hz$. That is why, only the frequency sent to the robot arms oscillators are divided by two. Their movements are then totally controlled. The oscillators of the tempo adaptation are kept the same, to avoid unnecessary perturbation.

A danger appears then when the adapted frequency, after being higher than $2Hz$, decreases again. For the robot, this means that the frequency would increase from oscillations around $1Hz$ to oscillations a bit less than $2Hz$. In this case, the difference of the previous sent frequency and the actual one is computed. The three same steps as before are implemented. If the difference is higher than $0.8Hz$, first, half of the actual frequency is sent to the robot. Then, an intermediate value is computed: a value between the previous sent value and the actual adapted value. And finally the actual value of the adapted frequency is sent to the robot oscillators.

The results are shown, in figure 5.4, on the left, with a function where the tempo is changed from $1Hz$ to $2Hz$ after 30 seconds and then, from $2Hz$ to $1.5Hz$. For the first tempo change, the sent frequency should be half of its value: it should oscillate around $1Hz$. For the second tempo change, the adapted frequency decreases from $2Hz$ to $1.5Hz$. The sent frequency should then increase from $1Hz$, that is half of the adapted frequency, to $1.5Hz$. In figure 5.4, on the right, the same limitation is done on the song by "System of a down". It can be noticed that the sent frequencies kept the same variations as the adapted frequencies.

5.3 RESULTS ON THE ROBOT

To see what these adapted frequencies really represented, some simulations were performed[3]. The results on the drumming task for a fixed tempo was already

[3]Some of the simulations done in Webots can be found on the website http://birg.epfl.ch/page67315.html

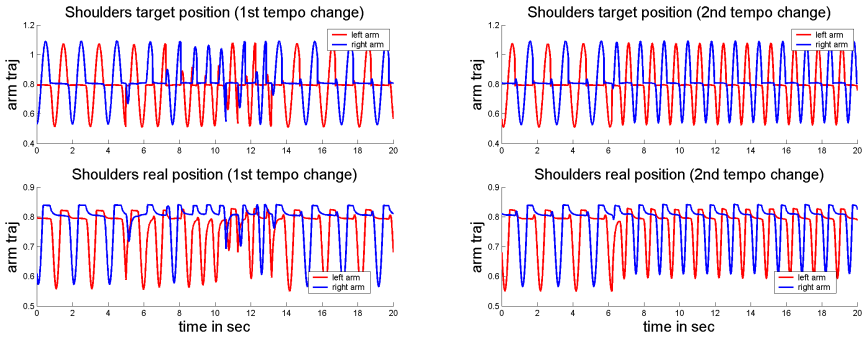


Figure 5.5: *behavior of the arm oscillators when a change of tempo happens five seconds after the beginning of the plots time scale. After a transient time, they converge again to the new music tempo.*

presented, in the previous section. Some another experiment is done here with the function that changes tempo. The difficulty of matching the arms movement with the beat pulses is shown, when watching the final results of the simulations.

5.3.1 OSCILLATORS BEHAVIOR

The behavior of the oscillators can be observed when a change in tempo appears. In figure 5.5, this happens after the fifth second of the plots. These are a zoom of the transition of the presented function whose impulses change from $1Hz$ to $2Hz$ for the first change and from $2Hz$ to $1.5Hz$ for the second change. Because the frequency limitation is implemented when using the robot, the first change will actually be from $1Hz$ to half of $2Hz$. It can be noticed that it took obviously more time for the first transition to get convergence. As the oscillators needed only 1 second to pass from half of $2Hz$ to $1.5Hz$, the first tempo change needed more time. In fact, the oscillators increased their frequency from $1Hz$ to $2Hz$, but with the frequency limitation, the robot oscillators had to converge to a frequency of $1Hz$. These changes explain the longer time taken to get convergence.

After both transitions, the robot hits the middle drum when the impulses were heard. These were then the concrete results for tempo detection, when implementing adaptive frequency oscillators. Some other simulations with songs followed and validated the human-like behavior of the robot, when listening to music.

Chapter 6 CONCLUSION

After a theoretical part on the oscillators and on music, a study was done on the feasibility of applying adaptive frequency oscillators for tempo detection of any song. More specifically, a tempo detection method using oscillators, which can adapt their frequency to an external input signal, was tested: it could be implemented in real-time, adapt to any changes in tempo and be robust to beat pulses that create perturbation and thus, an irregular tempo.

The advantage of using oscillators when detecting the rhythm of a song is their capability to synchronize. Their frequency becomes the tempo and their phase the moment of the "downbeat". Furthermore, the adaptive frequencies phase oscillates automatically around the external input signal phase, when adaptation is reached. This implies less computational time and it is more accurate for an implementation in real-time.

Unfortunately, like Scheirer's and Large's methods, the adaptive oscillators method developed in this project needed an onset detection. Its pre-processing steps are some of Scheirer's method. The computational time of the adaptive frequency oscillators method can be shortened by taking only part of Scheirer's steps. Moreover, like Scheirer's and Large's methods, the efficiency of this method depends on the initializations of the parameters and the initial conditions of the oscillators. Thus, if these are well fixed, the method allows a tempo detection of any kind of music, from simple impulse functions to classical, dance or electronic music: as long as they have periodic beat pulses.

Comparing the achieved results with the ones of the existing methods, it can be noticed that no exact tempo could be found for one specific song. Furthermore, integer multiple or integer fraction of the music tempo can be adapted. So which method is the most accurate? Which tempo is the most precise? Which are the most important parameters to be controlled when reproducing a "downbeat"? All these points lead to one question: what about the real perception of tempo? As written in [15], "there is no "ground truth" for rhythm to be found in simple measurements of an acoustic signal. So a method can not guarantee perfect detection and can not perfectly correspond to the intuitive sensitivity of a human listener." In the simplest case, a single periodicity is used when tapping along the rhythm of a song. But, people often move to much more complicated patterns as a simple periodic beat. Thus, it is hard to find a precise tempo. That is also an important point related in [8]: "except in the case of rhythmically simple music, it is rarely clear from a straightforward analysis of the acoustic signal where the beat is located, even though it may be quiet obvious upon auditory presentation of the musical material". As the study showed here, there are many different aspects that are linked to tempo detection of a song. Coordinated periodic movements as intuitively produced by

human beings are not an easy task to be recreated, even if dynamical systems used either in tempo detection or in the arm movements of a drumming task are well suited for such periodic behavior.

Bibliography

[1] P. Amstutz, *Semester Project: Synchronization of movements of a real humanoid robot with music*, February 2007.

[2] J. Buchli, L. Righetti, and A. J. Ijspeert, "A dynamical systems approach to learning: a frequency-adaptive hopper robot,"in *Proceedings of the VIIIth European Conference on Artificial Life ECAL 2005*, Lecture Notes in Artificial Intelligence, pp. 210–220, Springer Verlag, 2005.

[3] J. Buchli, L. Righetti, and A. J. Ijspeert, "Engineering Entrainment and Adaptation in Limit Cycle Systems. From biological inspiration to applications in robotics," in *Biological Cybernetics*, vol. 95, no. 6, pp. 645–664, 2006.

[4] S. Degallier, C. P. Santos, L. Righetti and A. J. Ijspeert, "Movement generation using dynamical systems: a humanoid robot performing a drumming task," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS06)*, 2006.

[5] M. Goto and Y. Muraoka, "Music understanding at the beat level: Real-time beat tracking for audio signals," in *Readings in Computational Auditory Scene Analysis*, D. Rosenthal and H. Okuno, Erlbaum, Mahwah, New Jersey, 1998.

[6] C. -C. J. Kuo and Y. Shiu, "Musical Beat Tracking with Phase-Locked-Loop (PLL) Technique," in *IEEE*, 2007.

[7] E. W. Large, "Modeling Beat Perception with a Nonlinear Oscillator," in *Working Notes of the IJCAI-95 Workshop on Artificial Intelligence and Music*, Montreal: IJCAII, pp. 24–31, 1995.

[8] E. W. Large, "On Synchronization Movements to Music," in *Human Movement Science*, vol. 19, pp. 527–566, 2000.

[9] E. W. Large and J. F. Kolen, "Resonance and the Perception of Musical Meter," in *Connection Science*, vol. 6, no. 2 and 3, pp. 177–208, 1994.

[10] B. Lewis and D. J. Berg, *Multithreaded programming with Pthreads*, Mountain View, California: Sun Microsystems Press, 1998.

[11] B. Nichols, D. Buttlar et al., *Pthreads programming*, Bonn[etc.]: O'Reilly, 1996.

[12] A. Pikovsky and Y. Maistrenko, *Synchronization: Theory and Application*, NATO Scientific Affairs Division, 2002. (II. Mathematics, Physics and Chemistry – Vol. 109).

[13] A. Pikovsky, M. Rosenblum and J. Kurths, *Synchronization: A Universal Concept in Nonlinear Sciences*, Cambridge University Press, Cambridge, 2001.

[14] L. Righetti, J. Buchli, and A. J. Ijspeert, "Dynamic hebbian learning in adaptive frequency oscillators," in *Physica D*, pp. 269–281, 2006.

[15] E. D. Scheirer, "Tempo and beat analysis of acoustic music signals," in *Journal of Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, January 1998.

[16] S. H. Strogatz, *Nonlinear Dynamics and Chaos*, Books Publishing, LLC, Cambridge, Massachusetts, 1994.

THANKS

* *To Sarah Degallier and to Ludovic Righetti, for assisting me in this project, for answering my questions and for correcting my reports.*

* *To Ludovic Righetti (again), for explaining me the theory about the oscillators and, especially about the adaptive frequency oscillators.*

* *To Alessandro Crespi, for introducing me into the theory of music, especially the different notions of sampling and filtering, for explaining me the details of programming and finally for his big help.*

* *To Auke Jan Ijspeert, for supervising this projet.*

* *And in general, to all people from the BIRG, for their presence and their support.*