Semester Project

Chaotic systems for escape and exploration in robots

Loïc Matthey

Under supervision of L. Righetti and Prof. A.J. Ijspeert.
Biologically Inspired Robotics Group

January 12, 2008

PAGE LEFT ON PURPOSE.

Abstract

This report presents the work accomplished during a Semester project at the **BIRG** Laboratory of EPFL. It took place in the Master winter semester of 2007-2008 at the EPFL.

We present a study of the applicability of Chaotic Systems for the control of robots. This approach to control being very new, very little is available as of now about its usability, usefulness or even robustness. But the first steps along the use of Chaos to control robots are very encouraging and promising. We implement a controller able to produce either a defined limit cycle behavior or chaotic exploration of movement, depending on the situation. We use feedback to stabilize the limit cycle behavior and provide information about the environment for the chaotic behavior.

We then study the advantages of using chaotic exploration in unknown terrains, compared to limit cycle behavior and pure random exploration. For that we perform a systematic benchmark of movement performance on uneven terrain of defined difficulties. We found out that a chaotic controller is able to move effectively on difficult uneven terrains, where a completely designed controller fails.

This work is inspired and strongly linked to the research of Prof. Yasuo Kuniyoshi, which is of the first people having introduced and studied chaotic systems for robot control.

PAGE LEFT ON PURPOSE.

List of Figures

List of Tables

PAGE LEFT ON PURPOSE.

Contents

Chapter 1

Introduction

1.1 Problem overview

Controlling robots is an old problem. It has been solved in many ways, and has given birth to a complete theoretical field: Control theory. But all this available theory was created and is applied in very well-known situations, where specific tasks had to be done. A typical example is the manufacturing of goods by industrial robotics. There is no place for unknown in general and the whole theory is build to produce strong and reliable behavior. To produce such a robust behavior, Control theory models and predict the unknown and the uncertainty for a given situation.

On the other side, the real world isn't well-known, isn't simple. Information changes all the time, noise is everywhere and unknown appears at every side-step. We can't model every cases. We need a theory that adapts well to a very changing environment, that can reacts to events and take new decisions if needed. We need something behaving like living things.

In our everyday life, we face changing environments all the time. We are able to move in this environment, modify our behavior accordingly, while still pursuing some goals.

Lots of different approaches on learning robust behavior have been proposed. They are usually divided in three categories :

1. Supervised learning.
2. Unsupervised learning.
3. Reinforcement learning.

In the first case, we want to learn something with the help of a "supervisor". This supervisor knows a-priori the answers, provides a measure of correctness for a given number of training examples. Applications of Supervised learning are for example "pattern classification". A typical pattern classification example is the recognition of hand-written digits.

In the second case, we just have the examples. We need to detect similarities among them, group them by those similarities (this is called clustering) and make sense of all these examples. This is a much harder problem, because we don't know what we want at the beginning. A typical example is the discovery of similarity in large database, for example buying habits on Amazon.

In the third case, we often have an agent, which can do several actions. Depending on what it has done, it can receive a reward. The agent tries to maximize its reward. A typical example is a mouse in a labyrinth, searching for cheese. The mouse has to search the space, and if it succeeds, will find the rewarding cheese. The problem is the lack of synchronization between actions and reward. Rewards comes very sparsely, and doesn't just depend on the action, but more on the global state of the agent in its environment.

Reinforcement learning is most similar to applications in robotics. Clearly, we want an agent to perform some task, with no prior knowledge of how to accomplish it, but with a notion of reward if the goal is reached. Such learning can be performed "online", that means directly while acting in the world, while doing trial and error. But this needs a lot of different trials before arriving to a good solution, and we want to avoid such long training time.

In nature, we can see young fawns stand on their legs just 20 minutes after being born [11]. This is not an easy task, yet they do this very quickly. It is thought now that the movement patterns are hard-coded into their spinal cord, so the only thing left to do is learn how to use the hard-coded elements. Still some adaptation has to be done very quickly.

We want to combine the two approaches, by learning quickly without doing so much trials. If we could rely on information that helped the learning, like the knowledge of our body or some pre-coded parts, then we could learn new behavior very quickly.

Chaotic Systems may prove useful in this way, because they allow a quick adaptation to the body, almost in real-time [22]. This makes them very interesting to produce really robust behavior that can adapt to deep changes in the environment in a few seconds. On the other hand, we can't store for now this information, so this isn't really "learning" yet. Moreover, we don't have much control on the obtained behavior, as it seems to emerge from body-environment interactions. But this is a first step using a completely different method, which may prove successful later on.

The use of chaos is more and more explored nowadays, as it seems to provides new ideas to solve existing problems. Moreover, studies have shown that sensory neurons of mammals have chaotic behavior, which is thought to be used to self-organize the information coming from the senses [12].

We want to combine existing control with the possible exploration power of chaos. Therefore, we want to design a system that can "switch" between **normal** and **chaotic** behaviors. The normal behavior is activated when the environment is known, when no problem arise. We can design such normal behavior very accurately, for example with non-linear oscillators via limit cycles ([16], [18]). But when the environment isn't smooth, when obstacles arise, such normal behaviors don't work anymore. So instead of taking care of every possible cases, we could switch the robot to chaotic behavior. We hope that such a chaotic behavior would take care of the uncertainty in the environment automatically. Then if the environment is safe again, we go to normal behavior again.

But before doing that, we first need to be sure that Chaos can actually bring something for control, and is not just a source of randomness. Our work will therefore concentrate on the following question:

- **Is Chaos more than just a Random movement ?**

To answer this question, we will implement a controller on a Robot. We will use a physics simulator called Webots to do this. The robot will be a Centipede Robot, developed at the BIRG Laboratory. We will then test the behavior of this robot using different kind of controllers, to see if the Chaos bring us something. Our main assumption is that the Chaos may be useful when the robot is on a unknown terrain. We will then test its behavior on such terrains, compared to more classical controllers.

We think that chaos can be very interesting and the research around it is quite new for robot control. It seems that this project takes place at a very promising time, and could provide useful insight. We hope it will.

1.2 Outline

This report is organized as follows: Chapter 2 presents the work of Y. Kuniyoshi and gives and overview of the theoretical notions used in this project. Chapter 3 introduce the Centipede robot and the work done by B. Jimenez on a locomotion pattern adapted to it. Chapter 4 shows the creation of our Rössler CPG and its mathematical investigation. Chapter 5 addresses the adaptation of the Rössler CPG

into a controller adapted to the Centipede robot, with a search of the optimal locomotion pattern. Chapter 6 shows the introduction of Sensory Feedback to our robot and its effect on our controller. Chapter 7 presents the creation of a Chaotic controller out of our Rössler CPG working in chaotic regime, and the optimization of its parameters. Chapter 8 introduces all the work we did to perform a systematic experiment to test the behavior of controllers on uneven terrain of growing difficulties. Furthermore, this chapter presents the conclusions that we could get out of these results. Chapter 9 concludes our work and present further possible work. Finally, Chapter 10 presents the additional material, the acknowledgements and the bibliography.

Chapter 2

Chaotic systems and control

As remarked in the introduction, robot control is a big field, with well established theory. On the other hand, while being studied mathematically for quite some time, and applied on weather models (which was the real example of chaotic behavior), chaos is less used in engineering fields. It is easily understandable: Chaos is hard to predict and control, which is exactly what we want for engineering problems. Nevertheless, we have theory on chaotic systems ([46]) and more interestingly for our scope, about their synchronization ([34]).

This chapter will present a small overview of the field and uses of chaos. We'll begin by what first triggered this project: the work of Y. Kuniyoshi on chaotic control of robots ([22] [21] [38] [37]). We'll then move toward more classical uses of chaos, and present some chaotic systems of interest for us.

2.1 Yasuo Kuniyoshi's work

2.1.1 Introducing the Coupled Chaotic Field

Y. Kuniyoshi first introduced his chaotic system model in a paper from 2004 [22].

He modeled a chaotic system with sensory feedback to dynamically produce movements adapted to the body. The goal was to be able to learn quickly how to move, by emergence of the rhythmic outputs adapted to the body.

This chaotic system is composed of an array of chaotic elements, following a standard logistic map (2.1), with a dependence on the sensory feedback provided by the body.

$$f_\alpha(x_i(t)) = 1 - \alpha x_i(t)^2 + \eta F_i(t) \tag{2.1}$$

The output of these elements are then combined to produce the signal driving the body (2.2). They are not directly coupled, but the coupling is thought to be carried out by the interaction between the body and the environment, carried out by the sensory feedback. See Figure 2.1 for a schema of this chaotic system.

$$x_i(t+1) = (1-\epsilon) f_\alpha(x_i(t)) + \frac{\epsilon}{N} \sum_{j=1}^{N} f_\alpha(x_j(t)) \tag{2.2}$$

He then tried his Coupled Chaotic Field, as it is called, on two kind of robots: one simple "muscle-joint" model (Figure 2.2) and an "multi-legged insect" model (2.3).

For the muscle-joint, they observed a back and forth movement on one axis. This axis could change over time, but it was pretty stable. When they modified the joint (for example added links), the movement changed, for example to a circle-like motion.

For the multi-legged robot, the robot started moving forward in a randomly chosen direction. This movement was stable and with a constant speed. They then modified the model to impose a goal to follow, by putting a weight on top of the robot (Figure 2.4). It allowed them to control the direction of

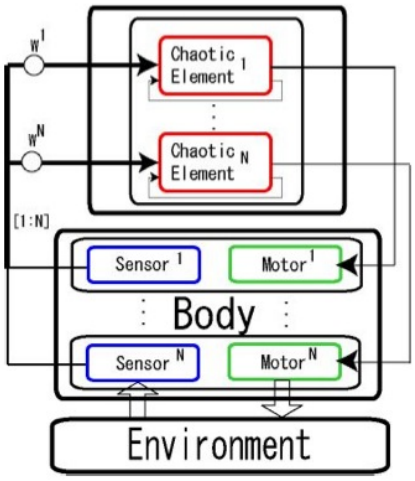


Figure 2.1: Coupled Chaotic Field of Y. Kuniyoshi. Taken from [37]

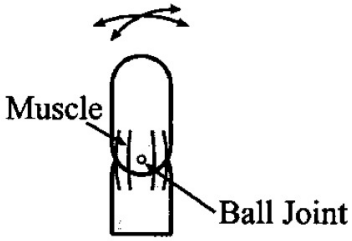


Figure 2.2: Muscle-joint robot of Y. Kuniyoshi. Taken from [22].

movement of the robot. All these movements emerged spontaneously due to the exploratory factor of chaotic elements. They then stabilized as the chaotic elements were brought to periodic behavior by the sensory feedback.

This is a very useful feature of chaotic elements: under certain conditions (e.g. external forcing) or parameters setting, they can have chaotic behavior or instead behave like nonlinear oscillators. The interesting question is then to know what can trigger this change. The first method to stabilize chaotic systems was proposed by Ott, Grebogi and Yorke (OGY) [29], by applying a specific periodic input to the chaotic system. Others methods exist to control chaos based on special input signals ([39]).

But the main problem with Kuniyoshi's approach is the lack of insight in the synchronization process and stabilization of the chaotic elements. Another EPFL student encountered problem when trying to reproduce the results from [22], as it seems that the system is highly dependent on the simulation parameters [4]. But this first work was very promising because the kind of movements produced were still possible and behaved nicely, which is impressive because the number of degrees of freedom was huge.

Y. Kuniyoshi and the members of his laboratory looked further into the possibilities offered by this new model. They tried incrementally to produce more and more theoretical insight on the learning and adaptation process.

2.1.2 Creating analyzing tools

The second paper on Kuniyoshi's Chaotic controller, published in 2005 by A. Pitti [37], introduced a new robot model. It consists of 10 prismatic elements connected by 10 force-controlled sliding joints. Each joint has 3 degrees of freedom. They are arranged in a ring-like fashion, creating some kind of

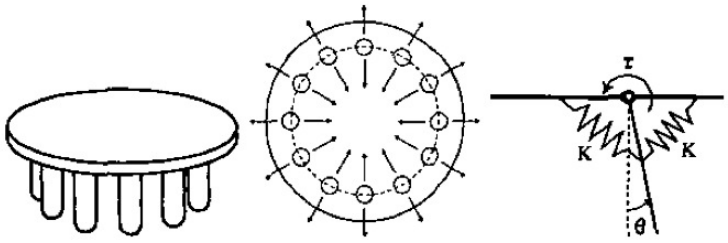


Figure 2.3: Multi-legged insect-like robot. Taken from [22]

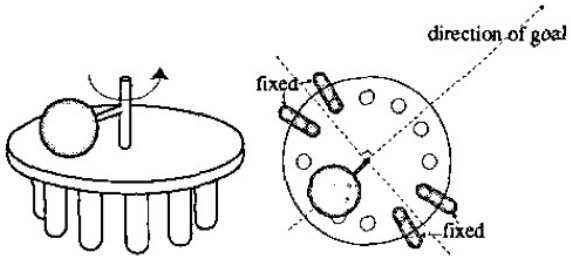


Figure 2.4: Multi-legged insect with goal control. Taken from [22]

flexible wheel (Figure 2.5). We see here that there are 30 DOF total. So figuring out a movement behavior isn't that trivial. They still use the Coupled Chaotic Field to study which movement emerged. Again the controller proved to produce interesting movement and behavior, adapting very smoothly to the body. But here the kind of movement obtained depends on the degree of chaoticity given to the chaotic elements. The robot could for example vibrate on place, or start moving forward then turn around and move forward for some more. It could also just rotate in equilibrium on some edge. So the obtained behavior was still way away from a complete controlled movement. It wasn't possible to control the behavior, and analyzing the synchronization and movement patterns was a hard problem. To try to gain insight in the movement patterns, they used three different tools to analyze the data: Spectral Bifurcation Diagram, Wavelet transform and Wavelet Bifurcation Diagram.

Spectral Bifurcation Diagram: This method has been introduced by Orrell and Smith in [27]. It offers another way of visualizing bifurcations for continuous-time systems. Bifurcations occurs when a parameter change produce a qualitative change of the system, for example period-doubling or becoming chaotic. Usual methods to visualize bifurcations like plotting the min and max values touched, don't show precisely the addition of harmonics and period of chaos, especially in high-dimensional systems. In order to obtain such a representation, the Spectral Bifurcation Diagram shows the amplitude power spectrum of the system, with respect to the parameter value. This allows to see brief periods of periodicity between chaotic periods more easily, as well as other bifurcations behavior. See bottom of Figure 2.6 for an example of a Spectral Bifurcation Diagram, taken from [27]. The chosen system is a 4-dimensional Lorenz'96 system. The Lorenz'96 system is a idealized one-dimensional model of the atmosphere, who can model the variation of the temperature, for example. It models a chosen "layers" of atmosphere, which control the dimensionality and precision of the model [27]. You can see the onset of chaos around $F = 12$ and the brief periodic windows at $F = 14.7$. On the same figure, on top, you can see a classical min-max diagram, which gives some indication about chaos and periodic behavior, but is clearly less precise than the Spectral Bifurcation Diagram. Moreover, the paper presents a 80-dimensional Lorenz'96 system, for which Spectral Bifurcation Diagram showed more than classical tools.

Wavelet Bifurcation Diagram This is a new tool introcuced by Pitti [37]. It observes the system with

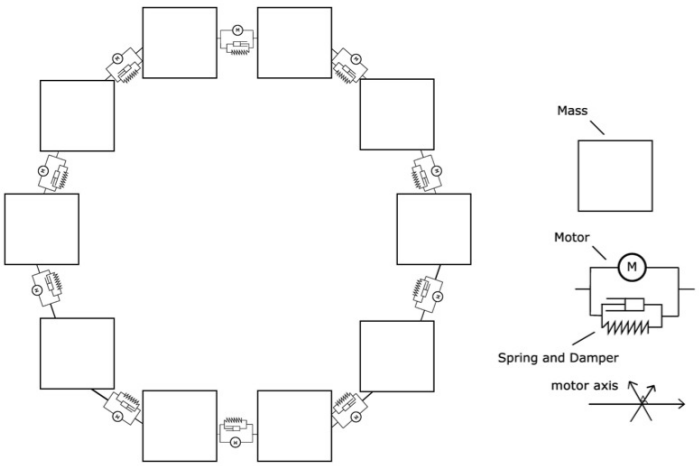


Figure 2.5: Wheel like robot model. Taken from [37].

respect to frequency, time and index of the neural unit. It is therefore able to see spatiotemporal changes, like the Wavelet Transform, but also to observe the behavior of neural units or groups of neural units, showing bifurcations in the system behavior (like the Spectral Bifurcation Diagram).

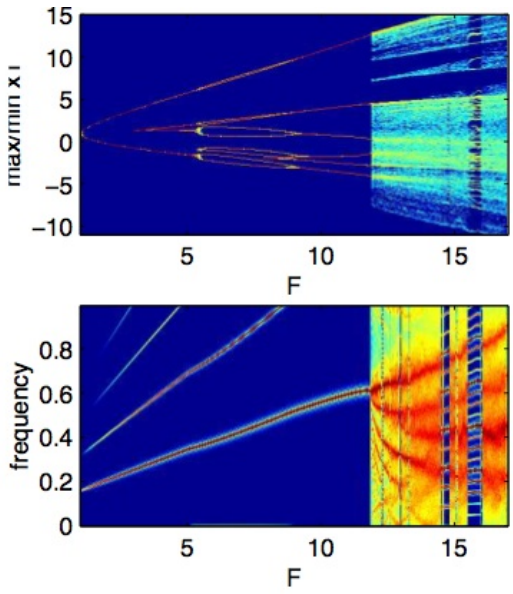


Figure 2.6: Min-max density and Spectral Bifurcation Diagram for a Lorenz '96 in 4D. Taken from [27].

They applied the Spectral Bifurcation Diagram on the body movement, providing information on the presence of coherent behavior. The Wavelet Transform and the Wavelet Bifurcation Diagram were applied to the output of the chaotic elements themselves, to detect whenever they showed temporal or spatial patterns. While providing some information, these tools weren't completely successful at capturing the interactions and synchronization that occurred between chaotic elements.

Continuing on the testing of the Coupled Chaotic Field, a third paper proposes yet another robot model and analyzing tools [38]. The robot proposed is a simple two dimensional bipedal robot. It again proved to generate interesting behavior, with walking, jumping and hopping movements. The

Spectral Bifurcation Diagram were used once again and this time provides really interesting inside on the dependance on the chaoticity degree of the elements. The Wavelet transform and the Wavelet Bifurcation Diagram showed a changing of scale of the stability of the dynamics when the movement became more complex. More precisely, when the movement is simple (like vibrating on spot), we see stability on low-scale study of the chaotic elements. On the other hand, when the movement is more complicated (like jumping), the low-scales show fractal patterns while high-scale are stable. This is in concordance with work on human locomotion [14].

We see that they tried to include some theoretical insight in the analysis of the results obtained with their Coupled Chaotic Field. Unfortunately, nothing is so much known about the real behavior of the chaotic elements. How do they synchronize and show periodic patterns ? Why do they produce these behaviors ? How important is the precision of the body model and the simulation ? What is the dependance between the sensory feedback and the behavior ? All these questions remain open.

2.1.3 From simple adaptation to motor development

After having conduct these experiments on different robots, one could easily foresee the possible applications of this new adapting model. The Coupled Chaotic Fields seems to capture quite well and really quickly the possibilities offered by the body. So we could try to use this capability to reproduce the learning occurring in Nature for every moving animal that has to use his body. This is called Motor development.

This is what Y. Kuniyoshi and S. Sangawa did in 2006 [21]: trying to model the learning of motor primitives. For this work, they took a strong inspiration from biological data and organization of the neuro-musculo-skeletal system.

They were able to see a spontaneous exploration of motor patterns, as well as some reduction of the degrees of freedom when the learning cortical model was present. This is very promising compared to known biological facts. They then implemented that learning model on a baby model, which showed nice preliminary results.

The important point here is that Chaos allows for quick and complete (in some sense) research of the posture/movement space. Moreover, the chaotic elements synchronize themselves automatically thanks to sensory feedback. But it is clear that a lot more insight in the inside process is needed before being able to really use such systems for more complex cases.

2.2 Other uses of chaos for robot control

When searching for other related work using chaos to control robot, we were quite surprised: there is nearly nothing else done, apart from Y. Kuniyoshi research. Chaos isn't really used for robots, maybe the field is too new, or the difficulties of controlling such emergent behaviors are too big.

We found an interesting application of chaos for perception-based navigation [3]. In this work, they use another kind of chaotic systems: Multiscroll system. This system consist of several cycle-like trajectory, aligned on an grid. The system can be driven to go trough certain scrolls only, creating recognizable patterns. They then used these patterns to represent the perception of the environment by the agent. While interesting, this work doesn't bring so much, as it's too linked with "common" robot control. Then didn't linked to perception directly to behavior for example, but relied instead on some movement rules.

But in a new conference on self-adaptive and self-organizing systems (SASO 2007), a work has been presented that shows an interesting application of Y. Kuniyoshi's work [10]. Duran et al. applied the Coupled Chaotic Fields to the control of a mechanical eye. This eye had to pursuit a dark point, moving in front of him. The only input was the difference between the point position and the center of the field of view. The pursuit obtained seems very good, even with a goal-directed input like that. This needs to be looked further into, because it may provides a really good method to implement goal-oriented behavior.

It seems that the use of chaos in robot control is slowly starting to grow, so the timing of this work is good.

2.3 Chaos control and Anticontrol

When looking into the use of Chaos in other fields than robotics, we quickly discovered one of it's most studied application: chaos control. A very good overview of the uses of chaos is given in [24].

Chaos control is used when engineers have some dynamical systems that can behave chaotically under certain conditions. Usually, this is a dangerous situation and chaos should be removed as soon as possible. The whole field tries then to attenuate chaos or to give securities on the non-existence of chaotic patterns for a given dynamical system (e.g. [13]).

Chaos control can also correspond to the goal of blocking some chaotic system to a given stable periodicity. This is most similar to the use needed to control robots, as common coupled oscillators are used to generate known trajectories producing some interesting behavior or movement. We don't have here the learning part and adaptation to the body present in Y. Kuniyoshi work, which can be a good or a bad thing. But, even without these added features, these methods aren't perfect and depend strongly on the models and desired trajectories [2].

Another research direction about chaos is Chaos Anticontrol. This is completely different from what we saw before. With Anticontrol, we try to create chaos out of some non-chaotic system. Known non-linear systems can switch to chaotic mode with certain inputs (e.g. Rössler oscillators). People modeled Neural Networks that could behave chaotically [6]. The goal is to use and direct them while in this mode to realize some goal. Usually, we want to explore some space. Again this is comparable but still different from what we're trying to do, because we need to keep the obtained insight of the chaotic exploration and reuse it later.

2.4 Chaotic oscillators and attractors

Chaotic systems have been discovered and studied since 1900. Their analysis is said to have started with the work of Henri Poincaré, while studying the dynamics of three-body orbits. He showed that very complicated (chaotic in fact) orbits were possible depending on initial conditions. Later studies were carried out by other scientists, such as G. Birkhoff, M.L. Cartwright, J.E. Littlewood, S. Smale and Soviet mathematicians, like A. N. Kolmogorov. But these studies didn't provide a lot of insight in the dynamics of chaotic systems, and their contribution wasn't embraced by the scientific community. But as computers allowed to repeat simulations trials, chaotic behavior became clearer and easy to observe ([28]).

A very important contribution to the field of chaos theory was published by E. Lorenz in 1963. He studied a simplified set of three equations based on the dynamics of a fluid near the convection onset. This system is now called the "Lorenz system" and is one of the most understood chaotic system available ([43]).

The name "chaos" itself was introduced in 1975, by the work of Li and Yorke. Their paper, "Period Three Implies Chaos", was very influential and anchored the name in the scientific community. Based on the work of A. N. Sharkovski, they proved that if we can observe a loop of period three in a system, then there exist loops of every possible period, which correspond to the definition of chaoticity [43].

Since then, a lot of different chaotic system have been proposed and studied. We don't want to present all of them, but only consider two of them, which are interesting for our application.

2.4.1 Lorenz system

The famous Lorenz system, being so much studied, has to be considered for our controller. It is defined by the following state equations :

$$\begin{cases} \dot{x} &= \sigma(y-x) \\ \dot{y} &= x(\rho-z)-y \\ \dot{z} &= xy-\beta z \end{cases} \tag{2.3}$$

It produces a 3-dimensional attractor with a butterfly shape, see Figure 2.7. σ is called the "Prandth number", ρ the "Rayleigh number". $\sigma, \rho, \beta > 0$, but usually one fix $\sigma = 10$ and $\beta = \frac{8}{3}$, while varying ρ to produce different behaviors.

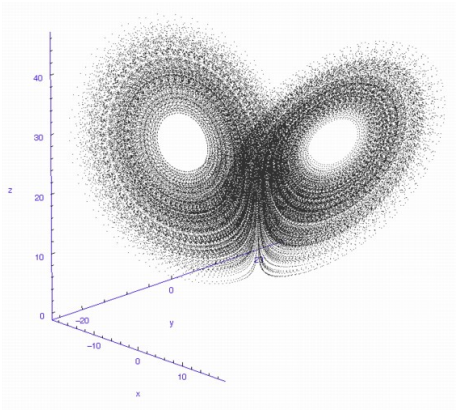


Figure 2.7: Lorenz strange attractor, taken from Lewis Dartnell, UCL

Small values of ρ make the system converge to a stable point, which is of no interest in our case. The typical value of $\rho = 28$ produces the chaotic butterfly shown before. Such a closed trajectory is interesting, but its 3D characteristic may be problematic, if we need to define some kind of phase. Bigger values for ρ, like $\rho = 400$, produces limit cycles, which could be useful, but still poses some problem for phase definition. See Fig 2.8.

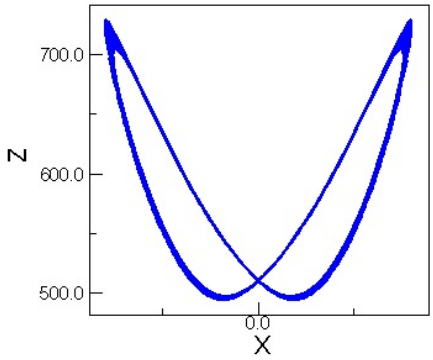


Figure 2.8: Lorenz limit cycle, for $\rho = 400$, taken from Michael Cross, Caltech

2.4.2 Rössler system

We consider another chaotic system, the Rössler system. It was build by Rössler in 1976 to exhibit the simplest possible strange attractor. Here is its description :

$$\begin{cases} \dot{x} &= -(y+z) \\ \dot{y} &= x+ay \\ \dot{z} &= b+z(x-c) \end{cases} \tag{2.4}$$

It only has one non-linearity, xz, and can produce interesting behavior. Its attractor is a simple cycle, with periodic "jumps", see Figure 2.9. These jumps introduce the chaoticity of the system. It can be put in non-chaotic mode, thus producing a smooth circle limit-cycle.

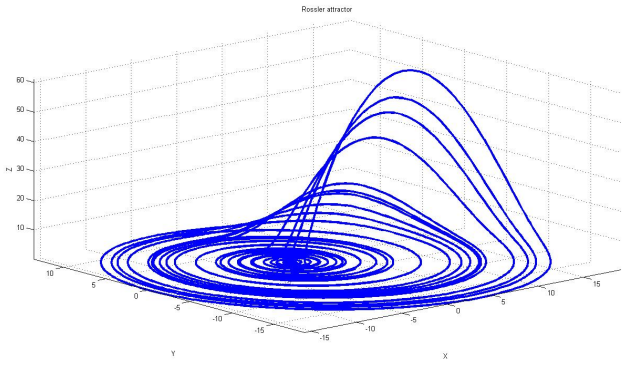


Figure 2.9: Attractor for Rössler System, chaotic mode, $a = 0.2$, $b = 0.2$, $c = 5.7$

If we fix a and c, b directly controls the chaoticity. Typical values are $a = 0.2$, $c = 5.7$ and b varied. With $b = 3$, we have a simple circular limit-cycle. When reducing b, we have several period-doubling bifurcations occurring, till chaos is reached at $b = 0.2$.

The fact that the attractor is pretty flat and close to a circle allows to define a simple approximation to the phase of the system. We simply project the trajectory onto the XY-plan, then calculate the phase like a circle :

$$\Theta = atan(\frac{y}{x}) \tag{2.5}$$

This is only an approximation, because when the system elevates from the XY-plan, we overestimate or underestimate the phase. But this is simpler than using the Hilbert transform (i.e. defining the phase as the argument of $s(t) + i \cdot H\{s(t)\}$). See Figure 2.10 for the projection on the XY-plan.

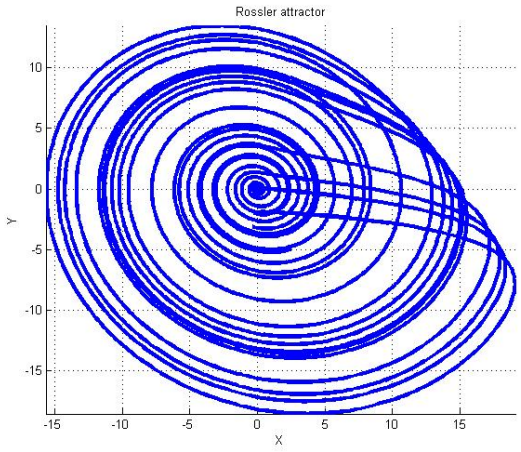


Figure 2.10: Projection onto XY-plan of the Rössler System in chaotic mode, $a = 0.2$, $b = 0.2$, $c = 5.7$

2.5 Synchronization of oscillators

We are going to use several oscillators for our controller, which need to be linked together to act as a whole system. This link is done by synchronizing the oscillators. Therefore, we need to define what we mean by synchronization and the different kind of synchronization possibles. This part is mainly inspired by the book on Synchronization by A. Pikovsky [34].

By Synchronization, we mean an "adjustment of rhythms of oscillating objects due to their weak interaction". In other words, we have two oscillating elements, say with frequencies f_1 and f_2, and via their interaction, their frequency detunning $\Delta f = f_1 - f_2$ will be modified. The oscillators produce a rhythmic signal, with an intensity of oscillation which is called their amplitude. We also need to define for each oscillator a phase. This phase depends on the time, and for a typical sine wave is something of the form $\phi(t) = \omega_0 t + \phi_0$. Now, we define the phase difference between two oscillators as $\Delta\phi(t) = \phi_i(t) - \phi_j(t)$.

Then we have the following definitions:

Phase locking: Phase locking occurs when we have $\Delta\phi(t) \approx constant$. Meaning that there is a relationship between the phases of two oscillators.

Phase synchronization: Such a synchronization occurs when two coupled oscillators achieve a certain phase locking. There is no constraints on the amplitudes of the oscillators. Such synchronization is interesting if we don't want to correlate the amplitudes of the oscillators, but we need to be able to define a phase.

Complete synchronization: Also called Amplitude synchronization. Occurs when both oscillators are synchronized to produce the same amplitudes at the same time. Simply speaking, we have $x_i(t) = x_j(t)$, i.e. a total synchronization between the oscillators. A complete synchronization implies a phase synchronization of 0, trivially. Such synchronization can be considered even when we cannot introduce a phase.

Such synchronization can be applied to non-linear dynamical system, even for system producing chaotic signals. It is therefore possible to synchronize two oscillators working in chaotic regime.

2.6 Central Pattern Generators

Another notion that we will need is the definition of a Central Pattern Generator, or CPG. A Central Pattern Generator is a "neural circuit that can produce a rhythmic motor pattern with no need for sensory feedback or descending control" [35].

In vertebrates, movement is generated by rhythmic electrical activities sent to muscles. Such rhythmic activities are generated by specific organization of neurons along the spinal cord. These groups of neurons are called Central Pattern Generators (CPG), as they are able to generate specific rhythmic output. The brain send different signals toward the spinal cord, called "drive signals", which can in turn activate certain area of the spinal cord, certain CPGs. The CPGs then send specific rhythmic output directly onto the muscles. It is thought that different groups of CPG generate different kind of movement, for example walking or trotting in quadrupeds.

As their biological existence is strongly assessed, people started building controllers based on the same principles.

A CPG in robot control is a single or a combination of oscillators, capable of producing rhythmic outputs. A very big range of oscillators can be used for this purpose (e.g. Hopf, Van der Pol), all we need is a dynamical system producing a limit-cycle behavior. If we use nonlinear dynamical systems, we can design the CPG in such a way that it will converge to a specific limit-cycle despite initial condition or perturbations. We obviously want that limit-cycle to correspond to a plausible locomotion pattern for our robot.

It is common to use one oscillator for every output desired, and then to link them, to couple them, to ensure a global predefined behavior. Such organization of oscillators are often represented as a graph. The nodes corresponds to the oscillators, while the label corresponds to the phase difference and coupling strength between them.

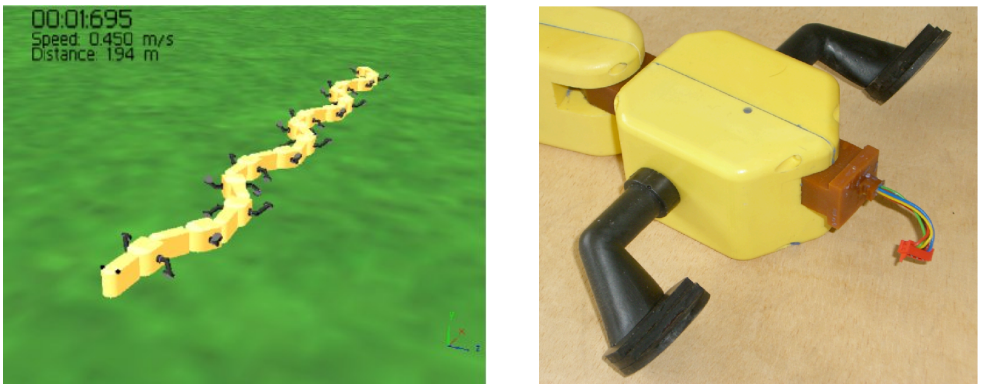
Chapter 3

Movement of a Centipede robot

For this project, we want to measure the applicability of chaotic system on a concrete robot platform. We used a new model of robot, which is an evolution of the previous ones developed at the Biologically Inspired Robotic Group of EPFL.

3.1 Earlier work

This robot is a Centipede-like robot, composed of a chosen number of body-leg segments. Each segment can articulate laterally to produce snake-like traveling waves. Moreover, to allow movement on uneven ground, movement is possible along the vertical axis with a spring joint. Each leg is independent and can turn around its axis. See Fig 3.1(a) and 3.1(b).



(a) Robot model in Webots

(b) Real body-leg segment

Figure 3.1: Centipede robot model and its hardware counterpart.

This robot uses the same hardware blocks as those used in the Salamander robot [19], which allows a easy construction of this new model. It has not been yet created in real hardware, so we use a simulation model instead. This simulation is done using the 3D physics simulator Webots©. It allows to get a realistic behavior, and in our case to incorporate the sensory feedback accurately.

A Webots model of the Centipede robot has been developed by B. Jimenez during his Master Thesis [20]. He studied the locomotion capabilities of the robot, optimizing controller parameters. He also studied movement on complex terrain, to test the robustness of the controller. He could obtain interesting behavior, which is a good point toward to use of this robot model for our experience on unknown terrain.

This also allows us to take example on the optimal locomotion pattern he obtained, and try to adapt it to our new controller.

3.2 Locomotion pattern

B. Jimenez tried several movement patterns, but we only consider the most complex one. The locomotion pattern is as follows :

1. Body segment oscillates according to the following equation :

$$f(k) = A_k sin(\omega t + \varphi + \Delta\varphi \cdot k)$$

   - A_k is the amplitude of oscillation.
   - ω is the intrinsic frequency.
   - φ is the phase difference between the body segment and its legs.
   - $\Delta\varphi$ is the phase difference between two consecutive body segment.
   - k is the index of the current body segment

   Depending on the $\Delta\varphi$, we can create a traveling or a standing wave on the body.

2. Body segment are connected between them via a spring-damped joint, that can move vertically. The torque on this joint follows the classical expression :

$$T = -\omega_{joint}\theta - \mu_{joint}\dot{\theta}$$

   - ω_{joint} is the spring constant.
   - μ_{joint} is the damping parameter.

3. Legs servos are directly positioned to the following angles:

$$\varphi_{right} = \omega t + \phi_r + \phi \cdot i$$

$$\varphi_{left} = \omega t + \phi_l + \phi \cdot i$$

   - ω is the intrinsic frequency.
   - ϕ_r and ϕ_l are the initial phase angles.
   - ϕ is the phase difference between two consecutive legs of the same side (just like $\Delta\varphi$ in the body).
   - i is the index of the leg.

To determine the optimal locomotion pattern, B. Jimenez optimized several parameters (A_k, ω, φ, $\Delta\varphi$, ω_{joint}, μ_{joint} and ϕ), with respect to the forward speed and capabilities of movement on uneven terrain.

See Tab. 3.1 for the obtained optimal parameters.

With these optimized parameters, we arrive at a plausible locomotion pattern: the legs touch the ground alternatively, following a descending traveling wave, while the body oscillates slowly in synchrony. See Figure 3.2 for an illustration of the obtained behavior (taken from B. Jimenez [20]). We'll try to mimic this behavior in our own system.

This kind of behavior is confirmed by a study on the scolopendra heros by Anderson et al. [1], See Figure 3.3. We can see in their paper that the body bend slowly following a traveling wave, with a much lower amplitude than our optimal locomotion pattern. We think this is most likely due to mechanical differences, as the real centipede is equipped with much more flexible and capable legs, compared to our rotating rigid legs.

Parameter	**Optimized value**
A_k (rad)	1.0
ω (Hz)	1.0
φ (rad)	0.0
$\Delta\varphi$ (rad)	$-\frac{\pi}{2}$
ϕ_r (rad)	$\frac{\pi}{2}$
ϕ_l (rad)	$\frac{3\pi}{2}$
ϕ (rad)	$-\frac{\pi}{2}$
ω_{spring} (Nm)	2.0
μ_{spring} (Nsm)	0.1

Table 3.1: Optimal parameters for Centipede locomotion, taken from B. Jimenez [20]

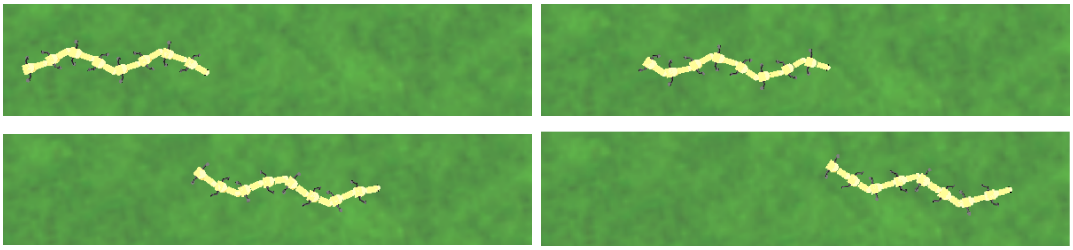


Figure 3.2: Illustration of the obtained locomotion pattern (Taken from B. Jimenez [20]).

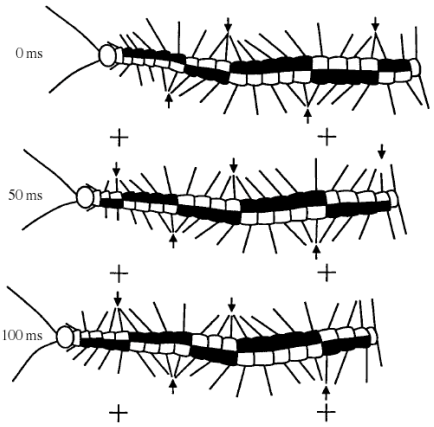


Figure 3.3: Motion of a real centipede, scolopendra heros (Taken from B. Anderson [1]).

Chapter 4

Rössler oscillators networks

As we already said before, what we want to create is a system that is capable of :

- Work in a completely fixed pre-defined fashion. For example its elements could follow specific limit-cycles, with known phase locking between different elements. Namely we want to design a Central Pattern Generator for robot control, like those used by Ijspeert et al. ([16], [18], [19]) and others.
- Switch to chaotic mode when desired. This mode will be used as exploration process and we hope will help to move the robot in unknown environment.

We thus need a mathematical oscillator capable of such dual behavior and that can be coupled, while following specific phase locking relations. We will then construct a CPG of those elements. The aim is to design the limit-cycle when the system is in "normal" mode, and then to switch the oscillators in chaotic mode when desired. We thus have complete control of the locomotion pattern when the conditions are good, with securities on the robustness and stability of the limit cycles obtained.

4.1 Rössler element

The first step to build our CPG is to choose the individual oscillators, which we call elements.

We presented in Section 2.4 several chaotic systems that can produced controlled oscillations: the Lorenz and Rössler systems. Both can be switched between normal and chaotic behavior by varying a parameter, and both can produce stable limit-cycle as well as chaotic behavior.

After a quick study, we compared the two elements with respect to our interests, see Tab 4.1.

They have both pros and cons, as we would expect. Important features here are the following :

Simplicity of behavior This is just a subjective measure of the obtained signals. In the case of the Rössler, it is very simple because it was designed to study chaos. The output of Rössler is just like

Criterion	Lorenz	Rössler
Scientifically studied	++	+
Simplicity of behavior	+	++
Numerical stability	+	-
Behavior plasticity	++	+
Limit-cycle capabilities	+	+
Phase definition	-	++
Mapping to robot inputs	-	++

Table 4.1: Comparison between Lorenz and Rössler element

a sinus wave in normal mode, and a sinus wave touching unknown extrema in chaotic mode. On the other hand, the Lorenz produced a more "sharp-knife" like pattern, switching between up and down patterns while in chaotic mode. Its normal mode looks like a biased sinus wave, it is not symmetrical and smooth.

Numerical Stability There is a disturbing problem with the Rössler system: when using too big couplings or certain parameters ranges, the system diverge toward $-\infty$ or ∞. This is problematic, but can be worked around if we know what are the possible ranges of parameters.

Behavior plasticity This represent the range of limit-cycles or possible solutions we can create with the systems. The Lorenz can produce equilibrium points, different chaotic attractors as well as different limit-cycles. On the other hand, the Rössler is more limited in the shape of its chaotic attractors and limit-cycles.

Phase definition This is a very important feature, we will present it in more details in Section 4.1.1. But we can already say that definition of a phase is very easy in the Rössler system, while being more tricky in the Lorenz.

Mapping to robot inputs This is a pure practical constraint: our robot takes as input servos and joint positions. If we can easily extract a periodic signal or a phase from the oscillators, this is easier to implement on the robot. The Rössler is quite good with that, as we have a near-sinus wave and a simple phase definition.

With respect to all these elements, we choose the **Rössler oscillator**. The easy mapping to robot inputs, its simplicity and phase definition are more important than possible behaviors of the Lorenz.

We use its common definition, see Equation (2.4), Section 2.4. We then use directly x as output, because it is the sinus-like signal in normal mode. We define the phase as the approximation given in Equation (2.5). This is only an approximation and can give artifacts when the projection onto the XY-plan isn't correct, but this is sufficient in our case.

4.1.1 The Phase problem

We already talk a bit about the difficulty of defining a phase in general for dynamical system. In the case of the Rössler system, on the contrary, this is quite easy. Taking only the projection onto the XY-plan of the states variables (namely using directly x and y to calculate a phase) gives plausible results, even in chaotic mode.

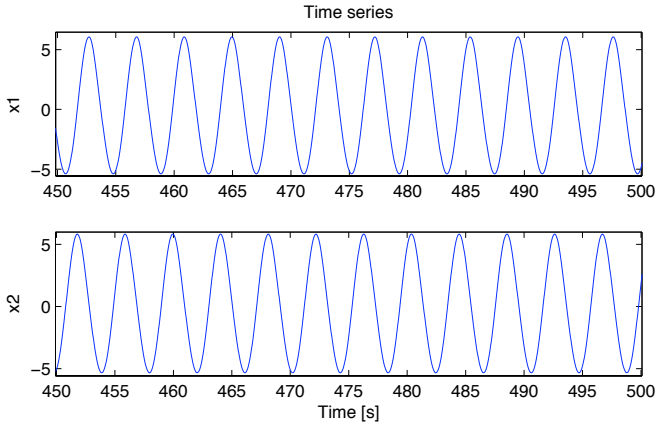
But we do get artifacts when we try to define a phase difference between two Rössler elements, as shown in Figure 4.1(b). These two elements are coupled with a phase difference of $\frac{\pi}{2}$. If we check on the time series, they are indeed very well synchronized (Figure 4.1(a)). But on the phase difference plot we observe some oscillations around the target value. These oscillations are due to the approximation of the phase, especially when the oscillator is in the first quadrant (see Figure 2.10). If we subtract two phase, one being in the first quadrant and the other being elsewhere, we will get a residue corresponding to the projection approximation. This isn't that bad, because it is only an artifact and does not penalize the synchronization of the oscillators. But we have to be aware of it.

We could overcome this artifact by defining a more correct phase, using Hilbert Transform. We will talk about that in Section 4.2.2.

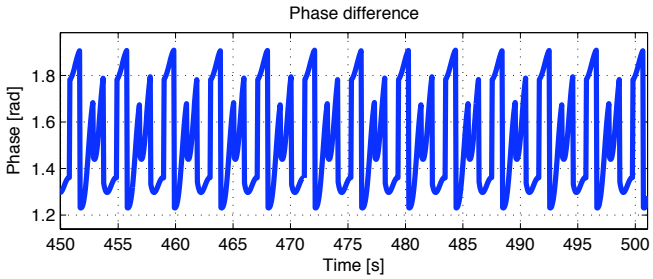
4.1.2 Modulating the frequency

For now, our Rössler element does not allow to tune the frequency of oscillations. This is an important feature in our case, because this allows to control the speed of locomotion.

We found a way of introducing frequency control in the Rössler element in [41]. It introduce a parameter ω to define the frequency, that acts as follows (Eq. 4.1):



(a) Time series, without artifacts



(b) Phase difference and its artifacts

Figure 4.1: Two coupled Rössler oscillators, with a phase difference of $\frac{\pi}{2}$. Illustration of phase approximation artifacts

$$\begin{cases} \dot{x} &= -\omega y - z \\ \dot{y} &= \omega x + ay \\ \dot{z} &= b + z(x - c) \end{cases} \tag{4.1}$$

The problem with this frequency control is that it breaks the balance between the parameters and the coupling. We had problems to create chaotic behavior, the parameter ranges had changed.

We thus use another way of controlling the frequency, shown in Eq. 4.2:

$$\begin{cases} \dot{x} &= \omega\big(-(y + z)\big) \\ \dot{y} &= \omega\big(x + ay\big) \\ \dot{z} &= \omega\big(b + z(x - c)\big) \end{cases} \tag{4.2}$$

This frequency control preserved better the relation between parameters and couplings, so we decided to use it.

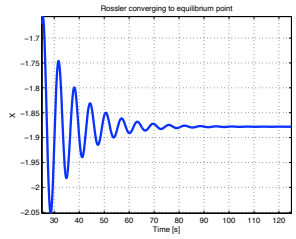
We are able to define a working frequency in Hertz for the element by using the following relation:

$$F_{elem} = \frac{2\pi}{\omega} \iff \omega = 2\pi \cdot F_{elem} \tag{4.3}$$

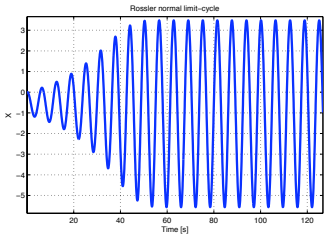
This comes from the fact that the frequency of an element with $\omega = 1$ is 2π. When multiplying by ω, we change the whole speed of the element linearly, thus giving this linear relation.

4.1.3 Behavior of Rössler element

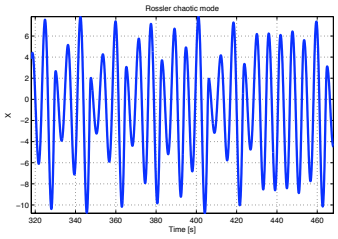
See Figure 4.2 for some time series created by the Rössler element.



(a) Convergence to equilibrium point. $b = 10$



(b) Sin-like limit-cycle. $b = 3$



(c) Chaotic mode. $b = 0.2$

Figure 4.2: Different time series obtained with a Rössler element, while varying parameter b. $a = 0.2$, $c = 5.7$

We already said that we chose b as a behavioral parameter. We use a value of 3 to work in "normal" mode, and we switch in "chaotic" with a value of 0.2. More complex behavior appears between these two values, such as 2-periodic, 4-periodic and so one limit cycles. This is easily shown using a Bifurcation Diagram. Such a graph can be found in Figure 4.3, for $a = 0.2$, $c = 5.7$ and while varying b. It shows the extrema touched by the x state variable. When we reach chaotic mode, there is an infinity of extrema touched, represented in this diagram as a vertical line. We see that for values of $b = 0.2$ or $b = 0.15$, we have a chaotic behavior, although it is not really a complete line, as the oscillator doesn't come near the origin.

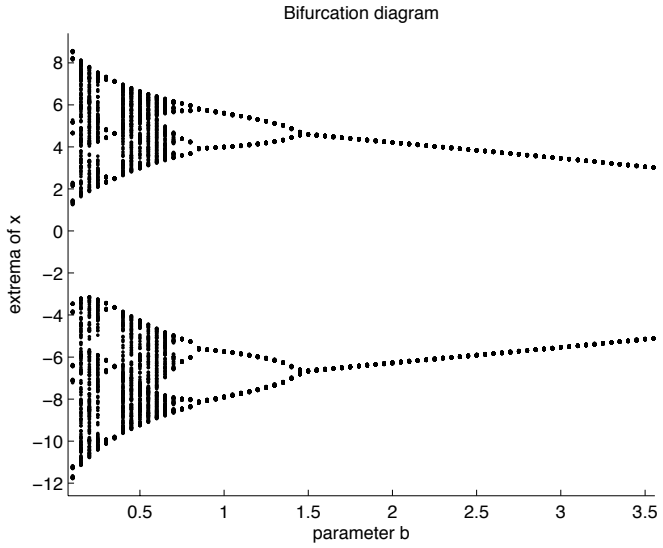
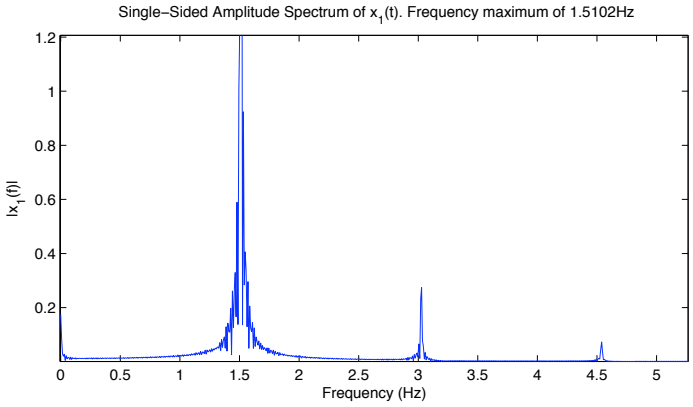


Figure 4.3: Bifurcation Diagram for Rössler system while varying b. $a = 0.2$, $c = 5.7$. Done with a Matlab script of J. Buchli.

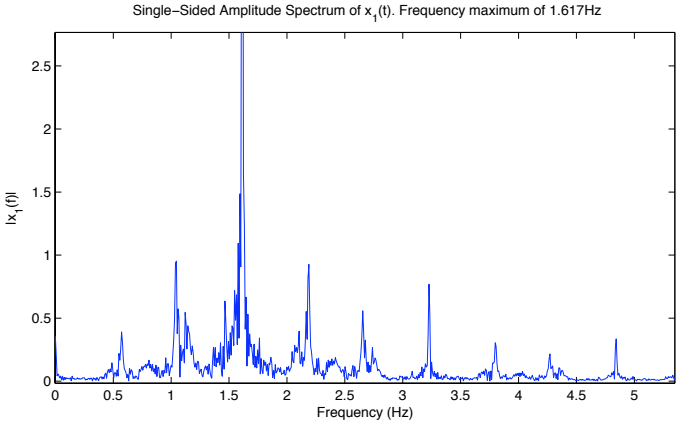
Frequency components

We can also study the frequency components of a Rössler element in limit cycle and in chaotic mode. We apply a Fast Fourier Transform to a 80 seconds long output of a Rössler element and show its Amplitude Spectrum (see Figure 4.4). The element has a defined frequency F_{elem} of $1.5Hz$, so with (4.3) we have $\omega = 3\pi$.

We see that in Limit cycle the frequencies are constrained to a neighborhood of the fundamental frequency and its harmonics. As desired, the fundamental frequency is around 1.5 Hz. We see that



(a) Limit-cycle mode



(b) Chaotic mode. $b = 3$

Figure 4.4: Amplitude Spectrum of a Rössler element. $\omega = 3\pi$

the amount of different frequencies overall are pretty small, which confirms the fact that our element is working in limit cycle mode with a fixed frequency.

On the other hand, in Chaotic mode, we observe several differences:

- There are much more frequencies covered by the element. We still see a "preferred" fundamental frequency, which is a bit higher than the desired 1.5 Hz (it is around 1.6 Hz). But we also see other frequency components, at 2.2 Hz and 2.6 Hz.
- The overall amount of frequencies is much higher too, which seems to show that the element touch a lot of different frequencies. This is normal because our component is chaotic.
- It is much more noisy, which again shows that the frequencies aren't that well defined.

We can conclude that our element in chaotic mode touches indeed more frequencies, which is a good thing. But it stays in a neighborhood of the desired frequency, which is not a good thing if we want the element to touch every frequency possible. But we are still visiting a lot of frequencies, and we can force the element to broaden its range of frequencies manually. A typical example is the addition of a random stimulation, which will drive the element away from different unstable cycles. But the Rössler element has a small positive Lyapunov ($\lambda = 0.0714$, compared to the one from Lorenz's system, $\lambda = 0.9056$ [45]), so we can not hope to generate every possible frequency. This discussion will make more sense when we will present the Observed Chaos Controller.

4.2 Coupling

Now that we have our element, we need to couple several together to build a CPG.

While looking in the literature, we found several people studying the effects of coupling with Rössler oscillators ([41] [42] [32] [15] [23] [25] [30] [33] [47]). Most of them wanted to achieve synchronization while being in chaotic mode. It was either complete synchronization (coupling the amplitude and phase), phase synchronization (coupling the phase) or lag synchronization (coupling the amplitude with a little delay between oscillators). Most of the time they try to couple system as different as possible, even to couple two completely different systems (like a hyperchaotic Rössler with a Lorenz).

This wasn't exactly what we needed. We needed a simple way to couple completely the Rössler system in normal limit-cycle mode, as well as the ability to define a specific phase difference between two oscillators.

4.2.1 Diffusive coupling

The common way of coupling two oscillators to induce a synchronization is to introduce a "difference coupling" [41]:

$$coupling(x_i, x_j) = K(x_j - x_i) \tag{4.4}$$

See Eq. (4.6)-(4.11) for an example. Such a coupling will reduce the distance between the two state variables coupled, converging to a state where they are both equals. In other words, we are trying to create a Complete Synchronization. The convergence time depends on the coupling factor K. We put this coupling on the x variable, following [41].

The general definition of a Rössler element with coupling is now:

$$\begin{cases} \dot{x_i} &= \omega\big(-(y_i + z_i) + coupling(x_i, x_j)\big) \\ \dot{y_i} &= \omega\big(x_i + a y_i\big) \\ \dot{z_i} &= \omega\big(b + z_i(x_i - c)\big) \end{cases} \tag{4.5}$$

Here is a simple example with two oscillators, as described in [41]:

$$\dot{x_1} = -(y_1 + z_1) + K(x_2 - x_1) \tag{4.6}$$
$$\dot{y_1} = x_1 + a_1 y_1 \tag{4.7}$$
$$\dot{z_1} = b_1 + z_1(x_1 - c_1) \tag{4.8}$$
$$\dot{x_2} = -(y_2 + z_2) \tag{4.9}$$
$$\dot{y_2} = x_2 + a_2 y_2 \tag{4.10}$$
$$\dot{z_2} = b_2 + z_2(x_2 - c_2) \tag{4.11}$$

In the example, the first oscillator will converge to the second one. Such a coupling achieves a complete synchronization exponentially fast in less than a cycle. Moreover, it is able to completely synchronize two coupled Rössler in chaotic mode. They are then producing the same chaotic pattern, with perfect synchrony.

This coupling is thus working well, but is too limited with respect to what we need to create our CPG: it can only induces phase difference of zero. We need to be able to design any phase difference possible. We did not find such a coupling in the considered literature, we thus have to create it.

4.2.2 Introducing arbitrary phase locking

We introduce a new coupling to be able to design any phase difference between two Rössler elements. We first project the states onto the XY-plan, for both elements. We have two referentials, one for each

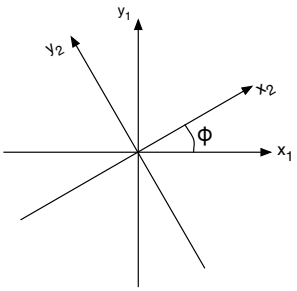


Figure 4.5: Rotation of the XY-plans, to keep a constant ϕ phase difference.

elements, which for now are the same. As the projection on the XY-plan looks like a circle, we can rotate one of the referential by a given angle ϕ. See Figure 4.5 for an illustration.

We then use a traditional diffusive coupling between the rotated state variables of the first element, and the original state variables of the second element. See Eq. 4.12. The distance between the rotated element and the other one will be reduced, thus imposing the defined phase difference. This comes only from the definition of the states and their projection.

$$\begin{bmatrix} x_\phi \\ y_\phi \end{bmatrix} = \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.12}$$

$$coupling(x_i, x_j) = K_{ij}(x_{j,\phi} - x_i) \tag{4.13}$$

$$coupling(y_i, y_j) = K_{ij}(y_{j,\phi} - y_i) \tag{4.14}$$

Following 4.12, we transform the coupling for the Rössler element as follows:

$$\dot{x_i} = -y_i - z_i + K\left(\begin{bmatrix} cos(\phi) & -sin(\phi) \end{bmatrix} \cdot \begin{bmatrix} x_j \\ y_j \end{bmatrix} - x_i\right) \tag{4.15}$$

Note that we do not use the coupling on $\dot{y}$, as it created instabilities and did not improve the convergence time. Moreover, as $\dot{y}$ contains x, the coupling will already be indirectly introduced on y.

Generalization to other elements

Such a phase difference coupling could be generalized to any other oscillator. But for that we need to define a phase plan fitting the system: that is, find a plan with $\dot{\phi} = 0$. Such a phase plan is very simple to define for the Rössler system, but is more tricky to define for other elements. More work is needed to determine the conditions of applicability of this new coupling scheme, but this could be an interesting and simple way to impose defined phase difference between coupled oscillators. Moreover, we can directly apply couplings defined for total synchronization, which are usually well studied.

4.3 Results

With the previous developments, we are now able to define Rössler elements, to couple them and define arbitrary phase difference locking between them. We are thus able to create arbitrary Rössler networks. Here are some results of convergence time and quality of phase locking.

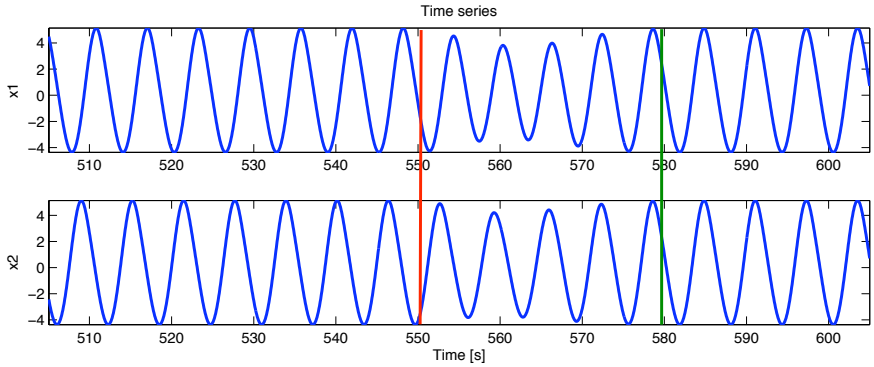
4.3.1 Limit cycle mode

We first let the two elements be in Limit cycle mode, that is, a periodic sin-like signal for the variable x. We put the parameters to $a = 0.2$, $b = 3.0$ and $c = 5.7$. The frequency parameter ω is kept to 1. Every figure has been created by an interactive Matlab script, available in the Matlab directory (rossler_coupled.m).

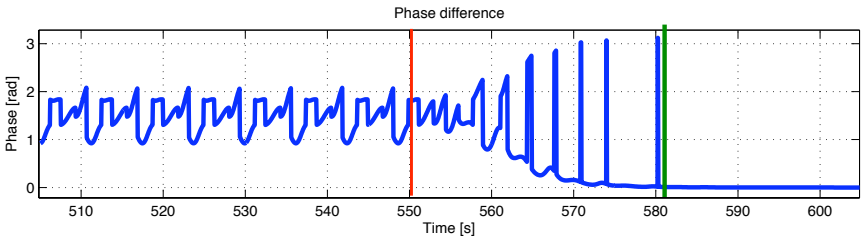
Total Synchronization We first try to attain Total Synchronization, that is, convergence to a phase difference of 0. See Figure 4.6 for the results. The first figure is the time series of the x variable for each element. At first they are randomly initialized, we indeed see that they are not synchronized. We put the coupling $K = 0.2$ at time $t = 550s$, marked by the red line. The green line mark the time of full synchronization. You can see on the Phase difference plot the exponential convergence towards Total Synchronization. The impulsions visible are due to phase slips of the oscillators. This is a characteristic of forced noisy oscillators, where periodic jumps of 2π can occur ([34], pages 81-83). In term of cycles, it takes approximately 3 cycles to converge. But this time can be made smaller by putting a coupling of $K = 0.4$, i.e. 1 cycle (figure not shown).

We need to be cautious with the coupling strength K. Indeed, we observed that a too big coupling cause stability problems and can even make one of the two element diverge toward $\pm\infty$. An empirical working range for the coupling strength K seems to be $[0, 0.5]$.

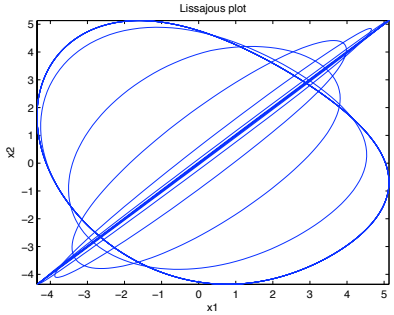
The last figure is a Lissajou plot, plotting one x variable against the other. The circle-like behavior was the state before putting the coupling on. Then we see the trajectory moving towards the diagonal, indicating Total Synchronization.



(a) Time series



(b) Phase difference. The oscillations are artifacts, the phase difference is constant.



(c) Lissajou plot

Figure 4.6: Synchronization of two Rössler elements to $\phi = 0$. Coupling $K = 0.2$ turned on at $t = 550s$.

Arbitrary phase Synchronization We then try to obtain two specific phase differences: $\frac{\pi}{2}$ and π. Every phase difference is possible, but we chose to illustrate those two. See Figure 4.7 for a phase difference of $\frac{\pi}{2}$. Here we indicated in purple the difference between the two time series. You can see that the phase difference of $\frac{\pi}{2}$ is indeed enforced, with a high stability.

On the phase difference plot, you see that the value of 1.57 is attained. The apparently periodic pattern around this value is due to the phase approximation we talked about in Section 4.1.1.

Again, the Lissajou plot shows this $\frac{\pi}{2}$ synchronization. We can remark that the Lissajou is not a perfect circle. That means that the output of the Rössler elements are not perfectly sinusoidal, which is again true because of the elevation on the z-axis (See Figure 4.8 for the 3D representation of one synchronized Rössler element).

Figure 4.9 shows the synchronization to a phase difference of π. Again the purple lines show the created difference. It's worthwhile to stress that this synchronization is harder to create than other phase differences. The sensibility to the coupling strength K is much higher, the elements can diverge really quickly if we put $K > 0.45$. It seems to come from the fact that this phase difference is the most different possible with the methodology we introduced. Little errors could increase rapidly, as the elements are not necessary in the same behavioral regions.

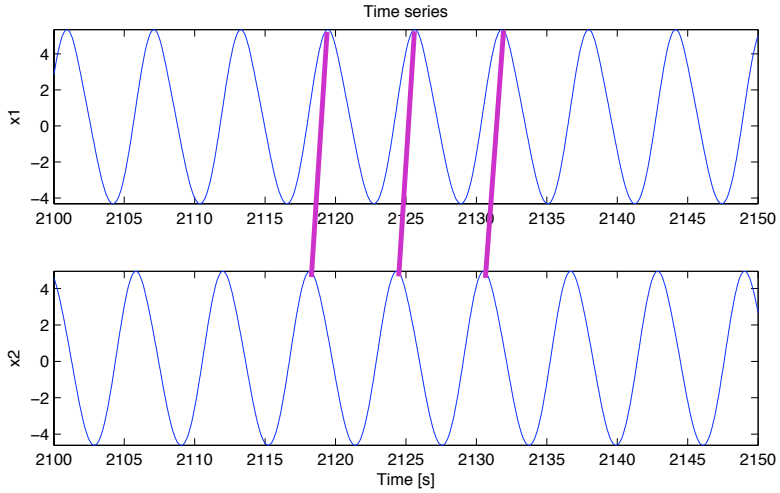
4.3.2 Chaotic mode

We then turn to chaotic mode, by changing the b parameter to 0.2. Using the same methodology as before, we try to produce Total Synchronization. The results are shown on Figure 4.10. Both elements are initialized randomly. The coupling strength K is set to 0.2 at time $t = 750s$, as marked by the red line. The green line shows the onset of perfect convergence. We see that Total Synchronization is indeed possible even if the elements are in chaotic mode, and while their states were completely different before the coupling. Before the red line, the time series have indeed nothing in common. Convergence is again exponential, with the same phase slips impulsions. It takes 5 pseudo-cycles. If you look on the time series, you can agree that synchronization is visually attained after just 1 cycle, around time $755s$.

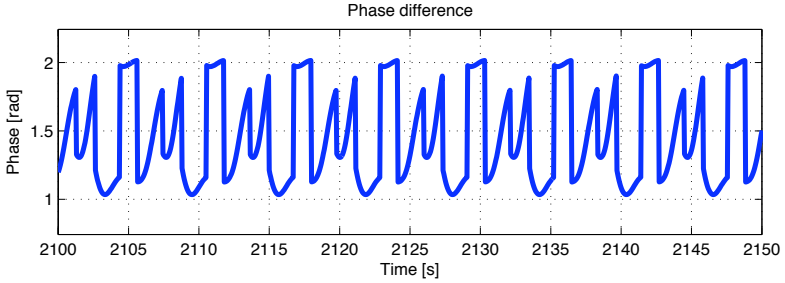
More importantly, we see that the patterns created are still chaotic, but they are completely synchronized.

We then tried to create other arbitrary phase difference synchronization. But unfortunately it does not work for too big phase differences. We can only produce phase difference from 0 to something around 1 radian. Bigger phase differences either simply do not work, produce strange behavior or even suppress chaos in one of the elements.

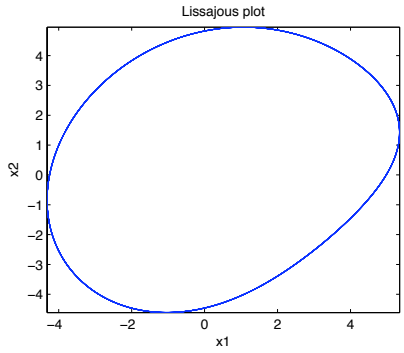
Still we can achieve synchronization for low values of ϕ. See Figure 4.11 for a so-called "Lag synchronization" of $\phi = 0.9$ rad. This is a nice feature, because people have proposed fairly complex methods to induce Lag Synchronization (e.g. [25], [42]), whereas our method is quite simple. Of course we are limited in the phases we can lock, and we still need to define a phase, which can be tricky depending on the systems.



(a) Time series



(b) Phase difference. The oscillations are artifacts, the phase difference is constant.



(c) Lissajou plot

Figure 4.7: Synchronization of two Rössler elements to $\phi = \frac{\pi}{2}$.

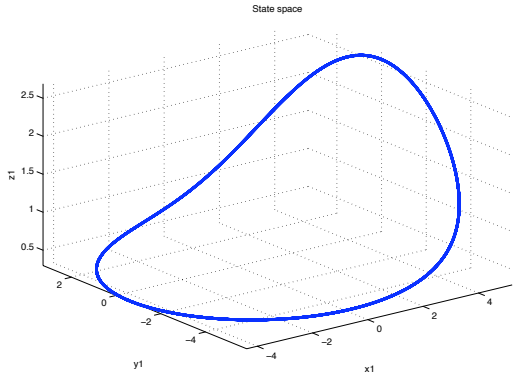


Figure 4.8: State space of a Rössler element in Limit cycle mode and synchronized with $\frac{\pi}{2}$.

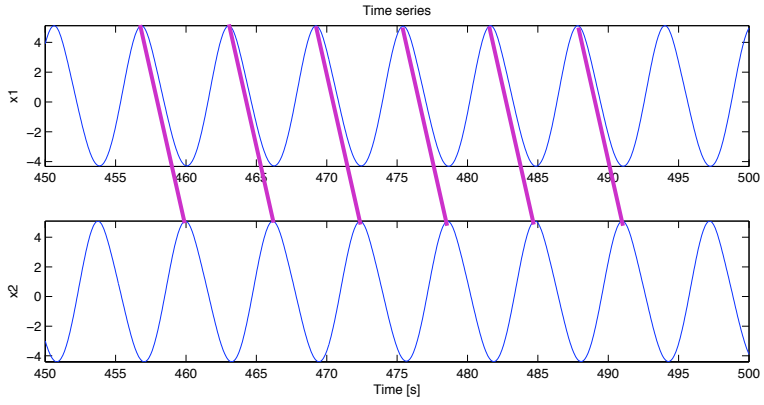
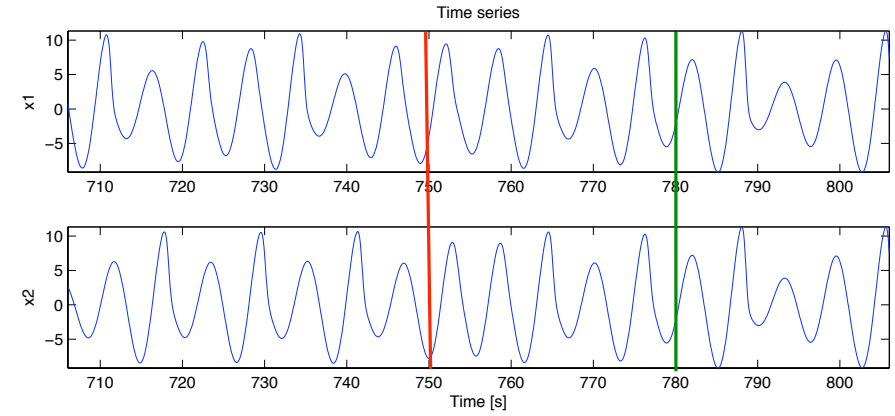
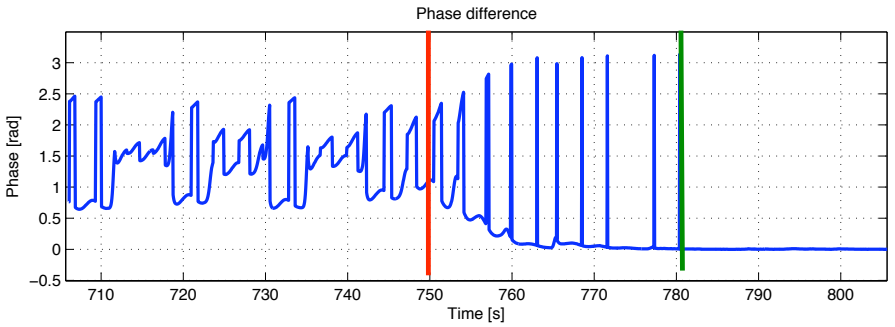


Figure 4.9: Synchronization of two Rössler elements to $\phi = \pi$.



(a) Time series



(b) Phase difference

Figure 4.10: Synchronization of two Rössler elements in chaotic mode to $\phi = 0$. Coupling $K = 0.2$ turned on at $t = 750s$.

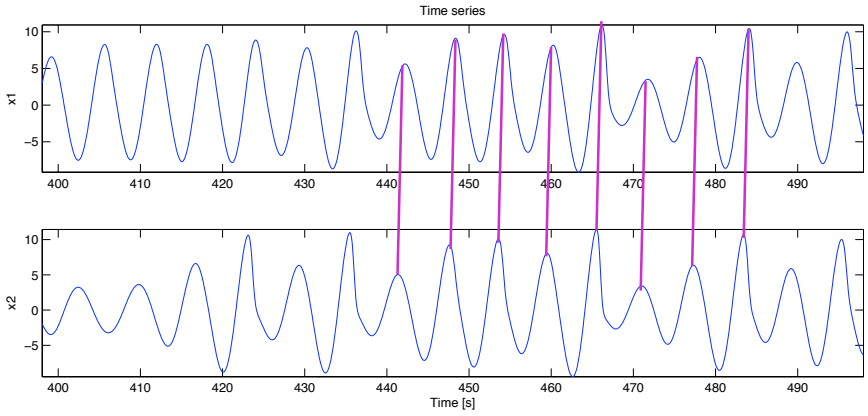


Figure 4.11: Lag Synchronization of two Rössler elements in chaotic mode to $\phi = 0.9$. Coupling turned on at $t = 435s$.

Chapter 5

Rössler network CPG for Centipede robot locomotion

Now that we have a Rössler network capable of creating arbitrary phase difference between Rössler elements, we can define a CPG for the locomotion of our Centipede robot.

We want to mimic the movement defined by B. Jimenez and presented in Section 3.2.

5.1 Architecture of network

We first need to transform the individual oscillators used by B. Jimenez into a equivalent CPG. This is easily done by looking at phase relations and possible maps of oscillators.

We use a Rössler element for every output, and couple them to introduce the desired phase differences. See Figure 5.1 for the obtained CPG in graphical representation. Every node is a Rössler element, and the arrows define the couplings, the labels corresponds to the phase differences between the elements. If we have a directed arrow $X \rightarrow Y$, it means that Y has a coupling term taking X into account. We chose to create a "spine-like" graph, to mimic biologic organization of elements. We call each triplet {Body, Left leg, Right leg} a "Segment". According to the optimization of B. Jimenez, each Body oscillator has a phase difference of $-\frac{\pi}{2}$ with its parent, and the legs should be in anti-phase.

This CPG has a minor difference with the locomotion pattern defined by B. Jimenez: the $\pi - \alpha$ and α couplings. In his implementation, he just put $\alpha = 0$. We want to stay more flexible and try maybe other values for α, so we leave it as a free parameter.

5.2 Theoretical results

We need to validate the ability of this CPG to synchronize precisely and quickly to the defined phase difference before being able to use it. The number of couplings is pretty high, so it could behave badly if the coupling strength is not correctly fixed. We also need to have a transient time (time before locked limit-cycle behavior) as small as possible.

We implemented this CPG in Matlab, and tested how well he produced synchronized output signals. We use 8 segments, meaning that we have 24 Rössler elements in total: 8 body element, 8 left legs and 8 right legs. They were all connected according to the mapping of Figure 5.1, with a coupling strength of $K = 0.5$.

5.2.1 Stability problems and adaptation

When testing the theoretical behavior of the Rössler CPG, we encountered several new stability problems:

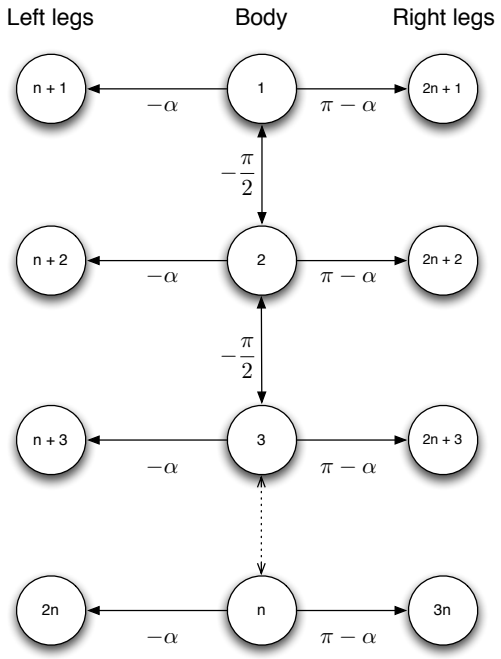


Figure 5.1: Rössler network CPG for Centipede locomotion pattern.

- The transient time before convergence was long, with big oscillations around the desired phase difference. The high number of Rössler elements and their interactions create new interaction flows, that seems to disturb the convergence.
- During the transient time, the amplitudes were varying a lot, possibly getting much bigger than the converged amplitudes. This would be very bad when mapping the elements output to the robot, because we create big oscillations at the beginning.

We thus adapt the CPG, by removing some of the couplings. In fact, we figured out that the coupling between a segment and its successor is not necessary. We only need to propagate a traveling wave of the desired phase difference. We don't need that the first segment should be influenced by its successors. So instead of bilateral coupling between Body elements, we put unilateral couplings. See Figure 5.2 for the new Rössler CPG network.

With this new network, we can achieve the same coupling, but interestingly with added advantages:

1. The convergence time is quicker. It seems that the bilateral couplings were introducing perturbations due to the "chain" architecture of the CPG. Groups of desynchronized elements could compete with other elements, perturbing the whole system. A perturbation introduced in the system could travel along the chain of oscillators for a long period of time. With unilateral couplings, the convergence is propagated from the top segment to the bottom one.
2. We can use bigger coupling strength. We do not have the same divergence problems that we had before, or at least they come only with bigger coupling strength. The new range of the coupling strength was found empirically to be near $[0, 1]$. This is also likely to be a consequence of interactions between groups of elements, that could make each other diverge. With unilateral couplings this is not possible anymore, and the only problem comes from a too big forcing that can make an element diverge.

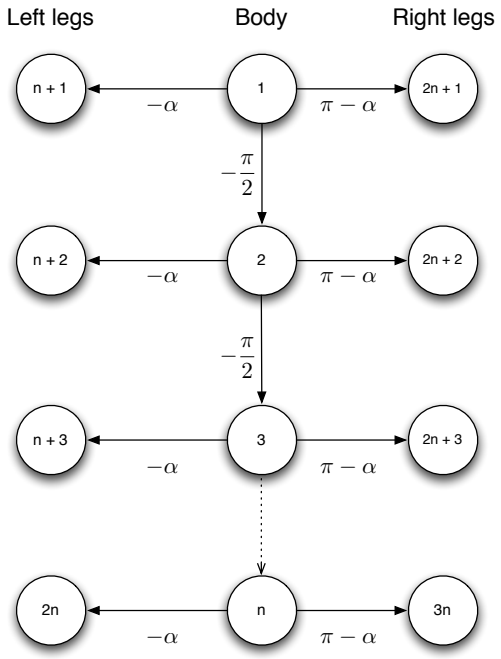


Figure 5.2: Adapted Rössler CPG network

5.2.2 Results

We will now present and comment some figures of convergence time and quality. Every figure has been created by an interactive Matlab script, available in the Matlab directory (rossler_movement.m).

Again, for Limit cycle behavior we use the classical parameters ($a = 0.2$, $b = 3.0$, $c = 5.7$), starting at random positions. We simulate the Rössler CPG shown in Figure 5.2, using $\alpha = 0$ and with a coupling strength $K = 0.5$ turned on after a while.

See Figure 5.3 for the time series. You can see all 24 elements, separated between Body (first box), Left legs (second box) and Right legs (third box). The actual time series are not that important, what is important is their synchronizations. We put the coupling on at time $t = 137s$, marked by a red line. Then the green line shows the time of complete convergence.

We can see that the convergence is very quick, in 3 cycles for the longest. This convergence can be better seen on Figure 5.4, showing the phase difference between the Body oscillator 1 and the Body oscillator 2. They should converge to $-\frac{\pi}{2}$ according to the design. Indeed we see this convergence, in an exponentially fast time. We also see in the time series that oscillators that are more "down" (that is, further away from Body oscillator 1) converge a bit later, in a delayed fashion. This is also normal because of the unilateral coupling we designed.

We also show on the time series the obtained traveling wave along the body, with the purple lines. It follows the phase difference of $\frac{\pi}{2}$ we designed. Moreover, the two other purple lines show the link between the Body oscillator and the Legs. With the Left we are completely synchronized with a phase difference of 0, and with the right ones we are in anti-phase (phase difference of π).

The last plot, on Figure 5.5, shows the Lissajou plot between the Body 1 and Body 2 oscillators. You see that they change from a near-synchronized behavior (near the diagonal on Lissajou plot) before activating the coupling, to a nicely synchronized state with $-\frac{\pi}{2}$ phase locking (the circle).

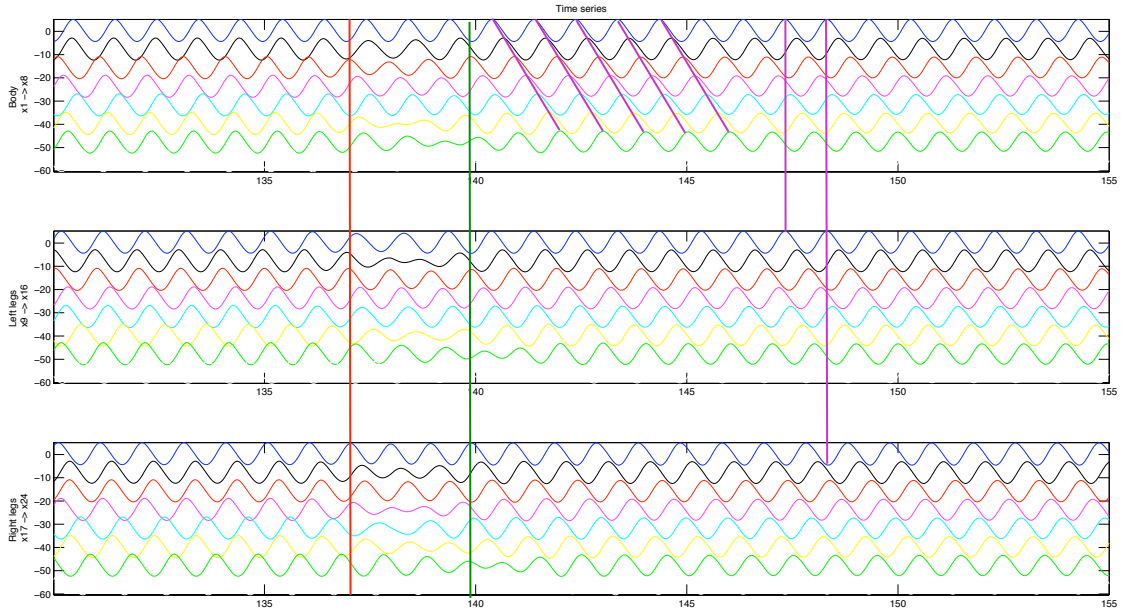


Figure 5.3: Timeseries for Rössler CPG for centipede movement.

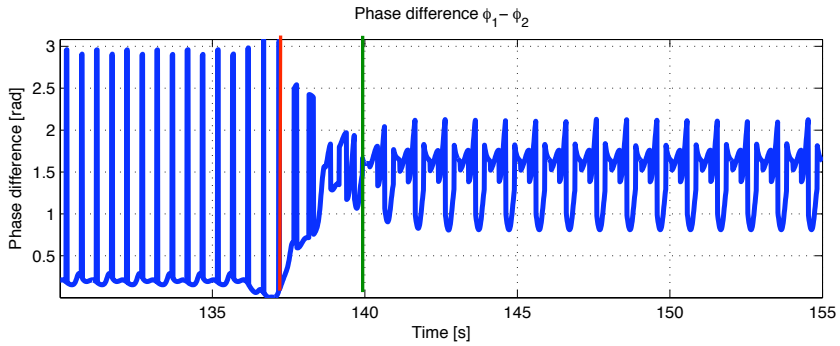


Figure 5.4: Phase difference between Body 1 and Body 2. The oscillations after the time $t = 750s$ are artifacts, the phase difference is constant.

5.3 Adaptation to robot

We have our theoretical controller, working all well in Matlab, but now we need to use it to move a real robot. We thus convert it to a Webots controller, using the robot model of a Centipede developed by B. Jimenez. Several difficulties arose during this conversion, we will present them quickly.

See Figure 5.6 for the mapping between the Rössler CPG and the robot parts. Let's go over the difference and difficulties we faced:

1. Our robot has a defined number of Segments of 8. So we have 16 legs in total. But because of its construction, it only has 7 Body joints. The last one being for the tail if we had one. So we have one exceeding Body oscillator, which we won't use. We chose to discard the last Body oscillator (the number 8). It is still present at the CPG level, because it is need for the last legs, but we simply don't map its output to the robot.

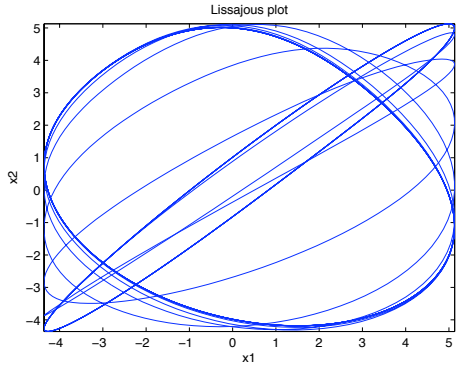


Figure 5.5: Lissajou plot between Body 1 and Body 2.

2. The Legs take as input a phase value, that needs to be always incremented. That is, the phase is not working modulo-2π. We then first have to convert our xyz-states into a modulo-2π phase variable, and then to convert this to a incremental phase.

3. The Body joints take as input an oscillating signal of a desired amplitude. We can therefore directly use the x variables of our Rössler elements, but special care has to be done with respect to its amplitude and mean.

4. The Legs have an initial phase, a bias. We can set it to any desired value. We can actually use this bias to replace the α parameter we put into our Rössler CPG. It does exactly the same thing, i.e. controlling at which moment of the cycle the leg is touching the ground. Therefore, we modify once again the Rössler CPG architecture and forget about this α term (Figure 5.6). This also allows a simplification of the coupling terms, because we only deal with 0 and π phase differences, which create nice trigonometric simplifications. So this is good because we need our controller to be as efficient as possible.

Let's see what parameters are present and how we took care of them.

Phase difference between body segments (fixed) This was taken from B. Jimenez work [20]. It is set as $-\frac{\pi}{2}$ and encompassed into the Rössler CPG.

Phase difference between legs This also comes from the work of B. Jimenez. We use the phase differences defined in the Rössler CPG architecture.

Body Amplitude (open) This is an open parameter controlling the gain of the Body oscillations. It needs to apply to a normalized signal, to avoid too big oscillations that could destroy the robot. The problem is our signals coming from the Rössler elements are not normalized. It is likely to be linked to the Frequency in its relation to movement speed.

Frequency (open) This is an open parameter controlling the speed of movement of every elements. This is directly controllable in Hertz, by using the relation with ω in the Rössler elements defined in Equation (4.3).

Bias of legs (optimized) This is an parameter that we optimize to create the good relation between the oscillation of the Body and the touching of the legs.

We thus explored the relation between the Body Amplitude and the Frequency, took care of the normalization and optimized the bias of the legs. All the following optimizations are performed in a flat empty world. We compute the distance travelled between the start position and the end position and use this as a metric. The simulation is run for 20 seconds. The number of runs averaged will be marked for every optimization.

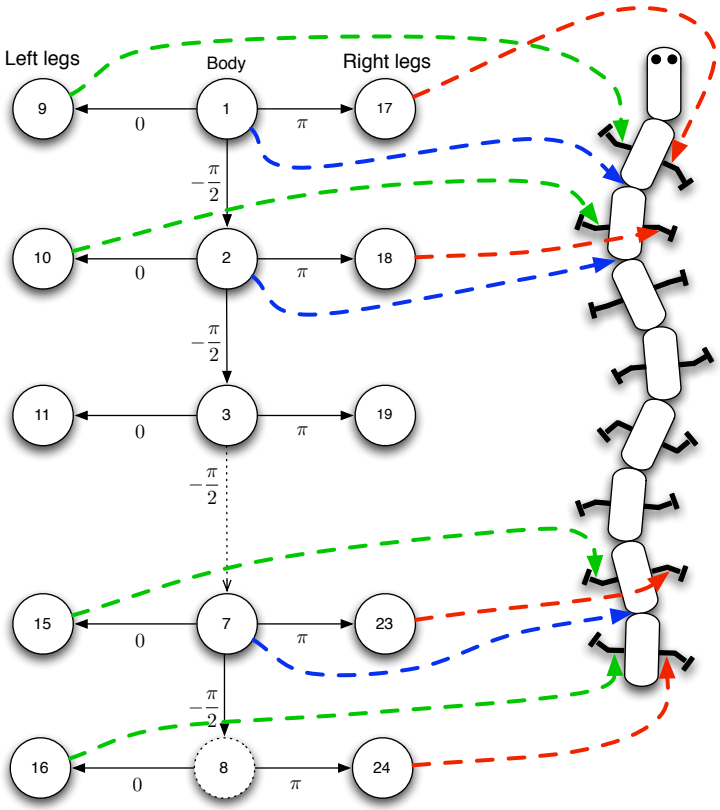


Figure 5.6: Mapping of the Rössler CPG architecture on the Centipede robot, dropping the α term

5.3.1 Normalization of outputs

A problem that we encountered while using directly the x variable of the Rössler elements to create the Body oscillation was that it was not normalized. The Body joints can take values in a restrained range, if we go outside the robot will be destroyed.

But the x variable amplitude is variable and actually depend on the parameters a, b and c used in the element. Moreover, we found out that during the transient time to convergence, this amplitude could increase temporarily to a bigger value than the converged amplitude. This is indeed a problem because we can't compute such transient amplitudes, as our initial points are random. Another thing is that the x value is not centered around the origin, but around a mean value that again depends on the Rössler parameters.

We decided to take care of these problems by dynamically normalizing the outputs of the Rössler elements before using them to control the robot.

This dynamic normalization works as follows, independently for every Rössler element:

1. We calculate a moving average on the x variable. This is done with this iterative moving average: $x_{avg}(t) = \beta \cdot x_{avg}(t-1) + (1-\beta) \cdot x(t)$. This β parameter controls the smoothness of the average. It was set to 0.95.

2. We calculate a moving max of the x variable. For that we keep a windows of W values, and calculate the maximum on this window. We actually do it in a more optimized manner (by checking the new value toward the max, and going through the window only when the actual max gets out of the window), see the code if you're interested. This W controls how much of the signal we consider when calculating the maximum. We need it to be big enough to capture one cycle, but not too big

if we want to adapt quickly to changes in the signal. For all our experiments, we use a window of size $W = 50$.

3. The moving max is initialized to a quite big value, to make sure that the robot doesn't oscillate too much at the beginning. This initialization was set to 12 in all our experiments.

4. Finally, we get a normalized value of the x-variable by using the following modified variable:

$$\tilde{x}(t) = \frac{x(t) - x_{avg}(t)}{x_{max}(t)}$$

The output of the Body joints is then directly:

$$y_{body}(t) = A_k \cdot \tilde{x}(t)$$

5.3.2 Optimization of touch of legs

At first, we ran our controller with an arbitrary bias for the legs of $bias_{leg} = -\frac{\pi}{2}$. This value was taken from B. Jimenez work[20] and then adapted visually because it gave a good locomotion pattern. We performed a systematic search of this parameter, setting the body amplitude to $A_k = 0.8$ and the frequency to $F = 2Hz$. Those values were chosen because they were quite safe in the locomotion pattern created. We average over 10 different runs, the Rössler elements are set to random initial conditions, using Limit Cycle parameters ($a = 0.2$, $b = 3.0$, $c = 5.7$).

See Figure 5.7 for the obtained optimization curve. We have a maximum at $bias_{leg} = 2.93$, which is just before π. The difference with our arbitrary bias is quite flagrant, we nearly double the performance!

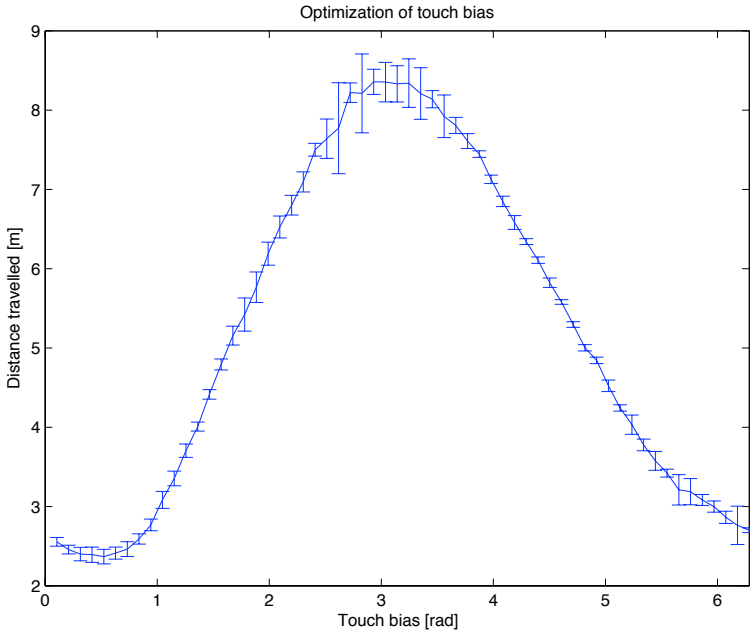


Figure 5.7: Optimization of Bias for the touch of the legs

This bias will be used later in the comparison, for what we call the Optimized Limit Cycle controller. We will also use the first non-optimal bias, for the Bad Limit Cycle controller.

5.3.3 Effect of amplitude of body and frequency

As those two parameters are the most open parameters we have to modulate the movement, we need to know how they are related to each other. Therefore, we perform a systematic search on the following range:

- Body amplitude, going from 0 to 1.8 rad.
- Frequency, going from 1.5 to 3.5 Hz.
- The Bias for the leg is set to the optimal value $bias_{leg} = 2.93$.
- Rössler parameters for Limit cycle.

We perform this simulation on 20 different runs and average the results. See Figure 5.8 for the results.

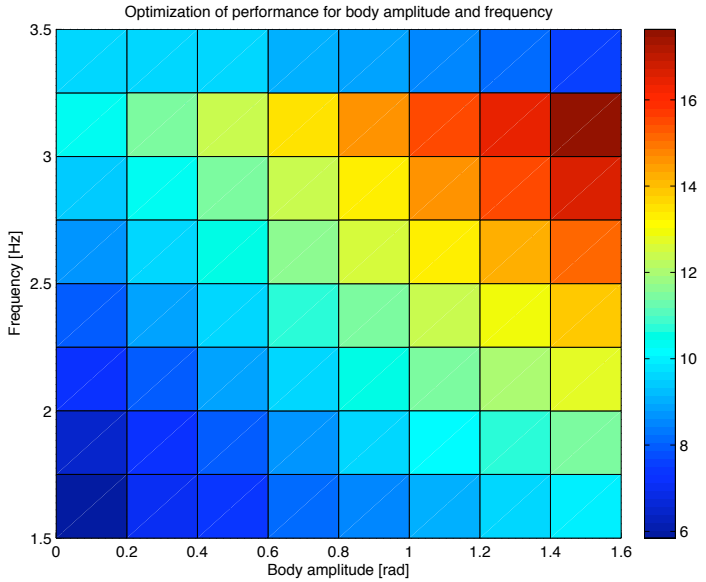


Figure 5.8: Dependance of performance on Body amplitude and Frequency

We see that the parameters are highly linked. There is a critical region for the Frequency above 3 Hz, where the performance drops dramatically. But everywhere else, there is a linear relation between the Body Amplitude, the Frequency and the distance travelled by the robot. There is a maximum value at $F = 3$ Hz, $A_k = 1.4$ rad, after which the performance degrades slowly. This is good news, because this means that we can use nearly every Body amplitude between 0 and 1.6, combined with any Frequency between 1.5 and 3 Hz, while ensuring good and reliable performances. At least this is the case on flat terrain.

5.3.4 Obtained movement discussion

We define two sets of parameters for the Rössler CPG working in Limit cycle mode, see Table 5.1.

We chose to keep an Optimized Limit Cycle working at 2 Hz, because this is the reference frequency we will use for our benchmark. For each Frequency, we take the optimal Body Amplitude at this frequency, according to Figure 5.8.

You can see the resulting movement in two different videos, one for the Optimized Limit Cycle, one for the Bad Limit Cycle. See in Additional Material.

The difference in attained speed is quite impressive, $0.9m/s$ versus $0.35m/s$. The movement from the Optimized Limit Cycle seems more natural too, more smooth. This is most likely due to the fact

	Optimized Limit Cycle	**Optimized Limit Cycle 2Hz**	**Bad Limit Cycle**
Body amplitude A_k	1.4	1.6	0.8
Frequency F	3.0	2.0	2.0
Bias leg $bias_{leg}$	2.93	2.93	$-\frac{\pi}{2}$

Table 5.1: Parameter sets for the Rössler CPG in Limit cycle mode

that, because the Bias of the legs is correct, every Leg helps in the movement. It means that every Leg touching the ground actually push the body forward, one after the other. That is not the case for the Bad Limit Cycle, where some legs have a bad timing with respect to body oscillations.

Chapter 6

Sensory Feedback

We now want to study what can Sensory Feedback bring to our controllers. More precisely, if we act in Chaotic mode, the Sensory Feedback will be the only possibility for the elements to interact indirectly. But first we need to find out what kind of Sensory Feedback we have at our disposition, and what we want to use it for.

6.1 Sensory feedback for the centipede robot

Our robot has no Sensory Feedback for now, so we will add it. We have the following feedback available:

1. Joint angles. We can know the actual angle of every joint. But as we are doing position control for our joints, this won't add any useful information. Indeed, we can assume that the actual position will be the one asked or very near them. Moreover, the current robot of the Laboratory that we plan to use does not dispose of this information. Therefore, we will not use that feedback.

2. Touch sensors. It is possible to get touch information for every feet of the robot. In this manner, we know when the legs actually hit the ground, and the applied pressure on everyone of them.

Therefore, we modify our Webots model to include Touch sensor on every feet of the robot. Then we can get a weight measure in kg at every timestep. See Figure 6.1 for an example of the kind of Sensory Feedback we get when we have a robot moving in Limit Cycle mode with a Frequency of $F = 1.5Hz$.

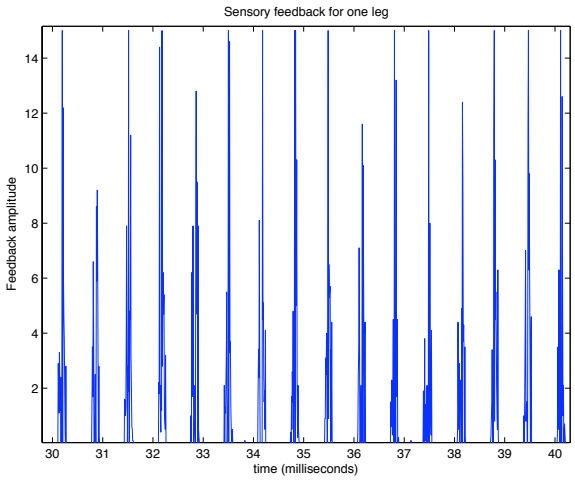


Figure 6.1: Sensory feedback given by the Touch Sensor of one leg when moving.

You see that we get a train of impulsions, at the frequency of the touching of the legs. This frequency depends directly on the movement pattern the robot is doing.

6.2 Theoretical implications

The reason we use Sensory Feedback is that we think it will help our controller in some way. The problem is we need to have some insight into what kind of help it can gives us.

Theoretically, we want to use the Feedback to adapt our controller to its body, or at least to change it a little bit according to the outside world interactions.

When looking at principles of synchronization, we can use a pulse train of a certain frequency Ω to try to synchronize the oscillator with this frequency Ω ([34] pages 62-65). This would imply a synchronization to the frequency of movement that automatically entrain itself, i.e. a resonant movement.

The other way to look at feedbacks is the modification of the dynamics of the system, making it go toward different attractors and region of its state space. It's then just a hope to think that these regions could be more adapted to the movement pattern. For example, if we act in chaotic mode, we could control the chaos, to make the oscillator work in a limit cycle mode, which is only defined by the feedback dynamics acting on it. Such chaos controls were studied by many people, but most of them use directly a time-delayed signal of the dynamical systems as feedback (see for example [9], which introduce a very nice way of controlling the chaos in the Rössler oscillators. Some of its conclusions are even generalizable to other uses than just chaos control). Other uses of feedback to control chaos can be seen in [39] (using an external oscillator or a delayed signals), [29] (time-delayed feedback), [47] (feedback and state observers) and a nice overview is done in [2]).

But we still need to know if the Sensory feedback we use can act as such time-delayed feedback. Considering the kind of signal that we have from our Touch sensor, it is unlikely that it encompass enough information to act as a time-delayed feedback. Therefore, we need to design more the action of our feedback, or simply use it to synchronize the controller to the frequency of its body, in a limited fashion.

6.3 Sensory feedback in the Rössler CPG

We need to use the Sensory feedback obtained from the Touch sensors into the controllers. We do this by modifying the Rössler elements, namely by adding a term to its system definition (Eq. (6.1))

$$\begin{cases} \dot{x_i} &= \omega\big(-(y_i + z_i) + coupling(x_i, x_j) + feedback_i\big) \\ \dot{y_i} &= \omega\big(x_i + ay_i\big) \\ \dot{z_i} &= \omega\big(b + z_i(x_i - c)\big) \end{cases} \tag{6.1}$$

We tested several feedbacks:

1. For the legs, we use two different feedback at the same time.
   (a) One depending on the touch signal of leg itself, its "ipsilateral" component, $touch_{ipsilateral}$.
   (b) Another one depending on the touch signal of the other leg on the same segment, its "contralateral" component, $touchcontralateral$.

   We combine them in the following way:

$$feedback_i = K_{fip} \cdot touch_{ipsilateral} + K_{fco} \cdot touch_{contralateral} \tag{6.2}$$

   The gain parameters K_{fip} and K_{fco} control the weight of the Sensory feedback desired. Note that they can be negative too. Then depending on the signs of those parameters, we create different behaviors, see Table 6.1 for their interpretation. But note that the last two don't make much sense from a locomotion point of view, so we just consider the first two ones. Furthermore, we put both parameters to same values, so we have $K_{fip} = -K_{fco} = K_f$.

sign of K_{fip}	sign of K_{fco}	Effect on current leg
+	-	Increase the stance time, with K_{fip} (slows down the leg when it touches the ground). Accelerate the leg when the contralateral leg touches the ground, thus shortening the swing time.
-	+	Shortens the stance time with K_{fip}. Slows down the leg when the contralateral leg touches the ground, increasing the swing time.
+	+	Increase the stance time, increase the swing time.
-	-	Shortens the stance time and shortens the swing time.

Table 6.1: Effect of Sensory Feedback on legs.

2. We also create a feedback that promotes the synchronization of the legs to a phase difference of π between them. This is already present in the Limit Cycle controller, but doesn't exist in the Chaotic mode. This did not produce interesting behavior, so we dropped it. The idea was to move the phase if the current touch feedback did not correspond to the predicted one from the current state space.

3. For the body, we tried to use several combinations of the ipsilateral and contralateral touch signals:

$$feedback_i = K_f \frac{touch_{ipsilateral} - touch_{contralateral}}{2}$$

$$feedback_i = K_f(touch_{ipsilateral} + touch_{contralateral} - max(touch_{ipsilateral}, touch_{contralateral}))$$

But neither of them were satisfactory, so we don't use them for now.

We end up using the simple feedbacks for the legs defined in Eq.(6.2).

6.3.1 Pre-processing of touch sensor signal

We pre-process the touch signals by using the following modified signal:

$$touch_{i,processed}(t) = touch_i(t) + \frac{touch_{i,processed}(t-1)}{2} \tag{6.3}$$

This pre-processing creates exponentially decreasing tails to the coupling signal. If you look back on Figure 6.1, you can see the raw touch signals. You see that it is quite impulsive and quick. Moreover, it has "holes" inside a region of high touch feedback, due to the bumping of legs. Such a signal can hardly do any effect on systems working at lower time-scale, so the addition of tails gives more time to the feedback to influence the system. See Figure 6.2 for the processed feedback signal.

6.4 Effect of feedback on performance

We performed a systematic search on the effect of this Sensory Feedback on flat terrain. We tested this feedback on two parameters set for the Rössler CPG in Limit cycle mode: the Bad Limit Cycle and the Optimized Limit Cycle 2 Hz. We use the defined parameter sets, and we perform this experiment on flat ground. We run 120 experiments per feedback value, each with random initializations for the Rössler CPG controller. Our metric is the distance travelled in 20 seconds. The coupling strength is varied from -0.5 to 0.5, testing the two first cases we discussed in Table 6.1. See Figure 6.5 for the optimization of the Bad Limit Cycle with Feedback, and Figure 6.3 for the optimization of Optimized Limit Cycle 2Hz with Feedback.

Interestingly, the results are quite different depending on the parameter set.

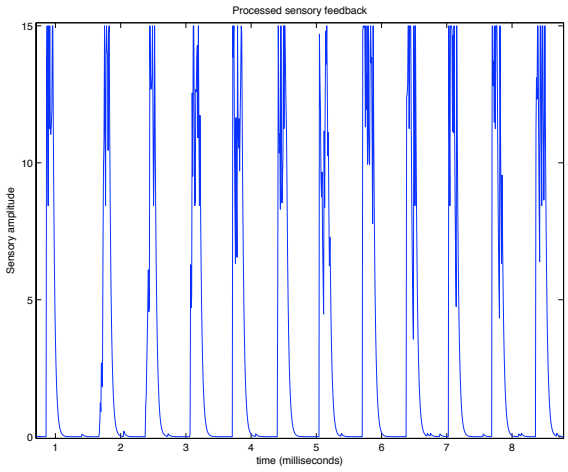


Figure 6.2: Sensory feedback processed to add tails.

6.4.1 Optimized Limit Cycle 2 Hz

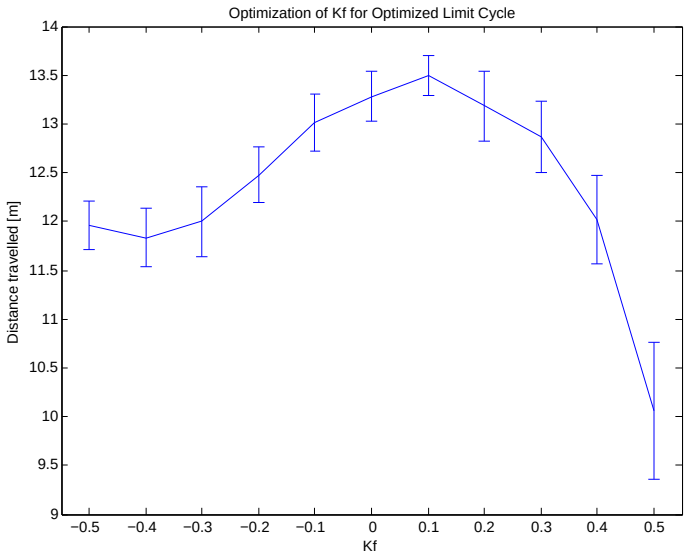


Figure 6.3: Effect of feedback parameter K_f on Optimized Limit Cycle 2Hz.

For the optimized parameter set, nearly every possible Sensory feedback strength decreases the performances. The only increasing feedback comes with $K_f = 0.1$, but the increase is nearly negligible with respect to the performance ratio (1.6% of increase).

This is understandable, as we already were at the optimal. This means that the sensory feedback can't adapt more the controller to the body that what we did with a systematic search. It either shows that we optimized a perfect controller for this robot, or that our feedback signal is not powerful enough. Maybe our feedback doesn't carry enough information about the actual body interactions to optimize the controller more.

Another nice way to look at this optimization is to plot the arrival positions of the different runs. We can then show what the parameter does on the trajectory, at least on a global point of view. See Figure 6.4 for this plot. Every parameter K_f is shown as a different color or marker style. The robot starts at position $(0, 0)$, and should move along the y axis. The graph is a bit messy, which shows

that the different parameter set don't change a lot the performance. We also see that the robot has a tendency to go toward the right, which could be due to the initial transient phase at the beginning, or because of the movement pattern. For a given parameter, there is quite a lot of dispersion in the arrival positions, but remember that the robot already travelled around to 12 meters, which is quite big. We see that the high positive K_f, which gave the worst results, are indeed stopped before the rest of the data points (light blue cross), and their dispersion is quite high. On the other hand, the points for no feedback (yellow-green circles) are confined to the top of the graph, with little dispersion.

So it seems that, indeed, using Sensory Feedback doesn't add a good thing for the already optimized parameter set.

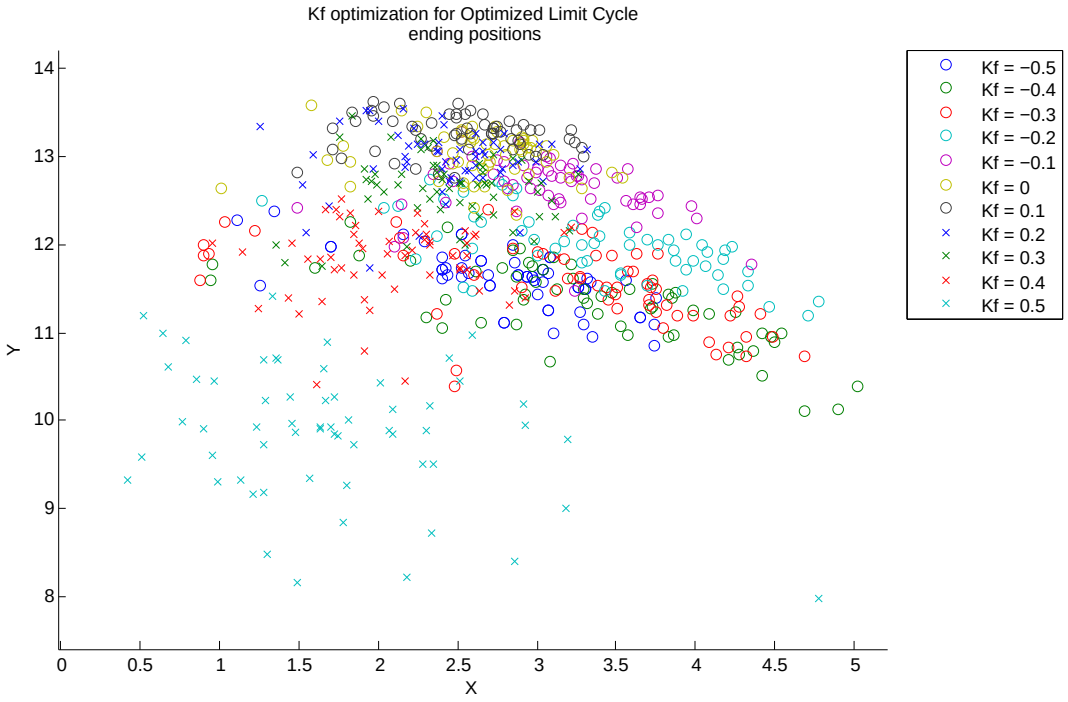


Figure 6.4: Final positions for different Sensory Feedback strength K_f for the Optimized Limit Cycle.

6.4.2 Bad Limit Cycle

For this parameter set, which is known to be sub-optimal, the Feedback actually increases the performance.

We performed a Kruskal-Wallis non-parametric test on these results, to detect if there really was an increase of performance. We obtained a P-value for the rejection of the Null-hypothesis that they were all generated by the same distribution with a confidence interval of 95% of 0. So there is an increase of performance. We also performed pairwise Kruskal-Wallis tests, using Matlab built-in function ("multcompare"). It allows to test the Null-hypothesis, with a confidence interval of 95%. See Figure 6.6 for the obtained confidence intervals for every Feedback strength.

The feedbacks go from -0.5 to 0.5, from left to right on the figure. You can see that the results are not statistically different for K_f that differ only by 0.1 (the intervals overlaps). But there is a statistical increase depending on the coupling strength. If we take the optimal strength of $K_f = 0.5$, we have a 27% increase of performance, which is significantly good.

Let's look again at the distribution of the final positions. See Figure 6.7 for the results, again with different colors for the different parameters. Here the picture is quite different than for the Optimized Limit Cycle. We clearly see "layers" of points, which correspond nicely to the arrival position for different Sensory feedback strength. Without any feedback (yellow-green circle), the performance is quite average,

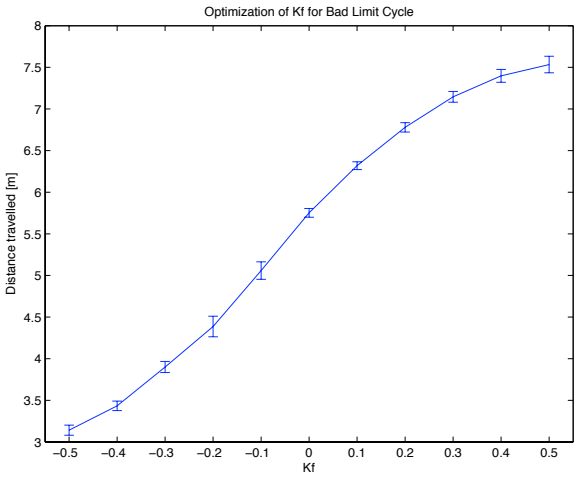


Figure 6.5: Effect of feedback parameter K_f on Bad Limit Cycle

arriving at $y = 6m$. These points show very small deviation in both direction. For growing Sensory Feedback strength (cross marker), we see a real increase in the distance travelled, but with an increase of the dispersion. It seems that the balance between both sides of legs is a bit modified when using a positive feedback, explaining the dispersion, but this promotes a bigger traveling speed, which is what we want. The case of $K_f = 0.1$ is quite interesting, as it increases only a little bit the performance but stabilize the movement a lot. For Sensory strength smaller than 0, we see a big decrease of the performance, as well as an increase in the dispersion. But this dispersion is nearly on a circle, so they could be due to change of direction at the beginning of the experiment. This feedback increase thus this initial dispersion, as well as slowing down the motion. This negative feedback shortens the stance time, and it seems that is a bad decision with the Limit Cycle.

So we can conclude that our Feedback indeed does something good for the robot movement. If we refer to Table 6.1, we see that it increases the Stance duration and decreases the Swing duration. It seems that letting the leg more time on the ground actually helps driving the robot forward. This was confirmed by a discussion with Prof. A. Ijspeert, who told us that increasing the Stance duration is a common way of increasing performances for this kind of robot.

But we also see that, even if it increases the performance, we don't get values in the order of the optimized controller. We are still far from that. Our feedback is therefore not powerful enough to fully optimize the motion pattern. We think this is because of the actual limitations in influence time available to the feedback. We already told that the Feedback consists more of impulsions, of short bursts. The feedback only act during those bursts, therefore they can't affect completely the controller as they should. Another way would be to link this feedback to the controller in another way. Maybe, as it seems to be a very important parameter for the Limit Cycle, we could link it to the bias of touch of legs explicitly. Or we could link it to the Body oscillators, trying to adapt them instead of the legs. This remains an open question.

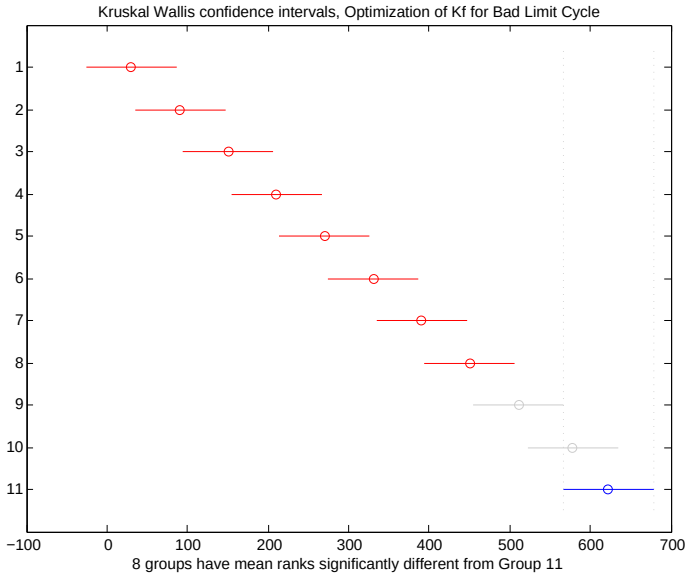


Figure 6.6: Confidence intervals for the different values of Sensory Feedback strength Kf on Bad Limit Cycle

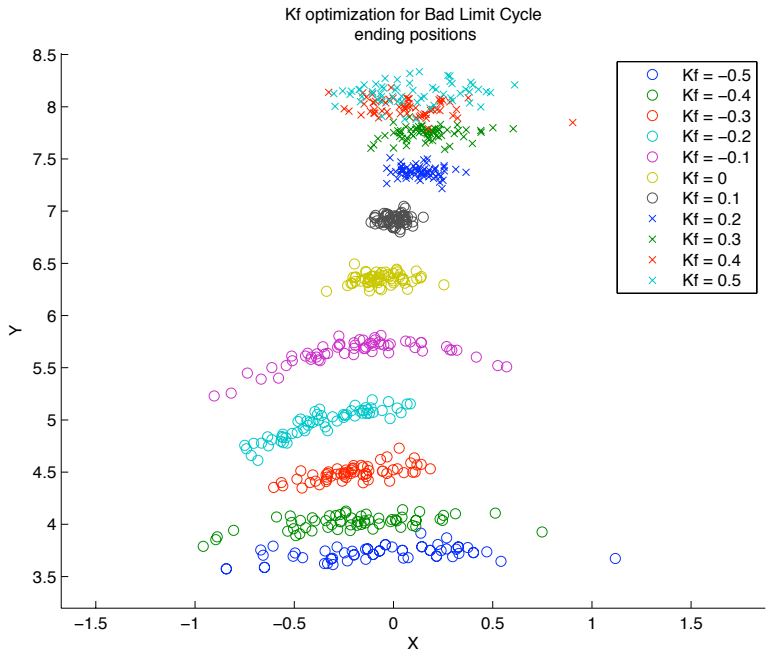


Figure 6.7: Final positions for different Sensory feedback strength K_f for the Bad Limit Cycle.

Chapter 7

Chaotic controller

We now turn ourself towards the actual use of Chaos to control our robot. We developed a Rössler CPG capable of generating a good locomotion pattern while acting in Limit cycle mode. But now we can make it chaotic just by changing one parameter.

We want to turn the controller into chaotic mode, while cutting away all couplings designed beforehand. We will now have independents Rössler elements working in chaotic regime. We then provide Sensory Feedback to those elements. We hope this feedback will take the role of the couplings, meaning that it will indirectly reflect the dependancy between elements by their interactions with the environment. This is one of the main ideas of Y. Kuniyoshi, which we think is interesting to study.

What is yet to investigate is:

1. Can those chaotic elements create a good locomotion pattern?
2. Is this locomotion pattern more robust on uneven terrain than a designed locomotion pattern?

As it is quite hard to really understand what relations exists between the Sensory Feedback, the elements and the chaotic behavior of the elements, we restrict our questioning to this simple question:

- **Is a chaotic controller doing more than a random controller ?**

We will thoroughly test this in the next chapter. For now we present the Chaotic Controller we developed, which is a modification of our Rössler CPG controller.

We then present a new theoretical conceptualization for a Chaotic controller: the **Observed Chaos controller**. This is a rethinking of the actual process taking place during movement, and an analysis of what chaos could actually offer us for locomotion creation.

7.1 Simple Chaotic controller

7.1.1 Behavior and parameters

We first construct a Simple Chaotic controller. That is, we simply modify the Rössler CPG controller to make it work in chaotic mode. This is simply done by setting $b = 0.2$ instead of $b = 3$ in Rössler elements. We also need to cut all the $coupling(x_i, x_j)$ terms we added to create the motion pattern. We want the controller to be completely free to explore its movement space.

We do use the Sensory feedback introduced in Chapter 6, in the exact same way as before. We hope that it will help the chaotic controller in some way to adapt itself to the body dynamics. This has been one of the things presented by Y. Kuniyoshi that seemed to work without having a real explanation. We found a possible explanation, using the new conceptualization presented later, see Section 7.3.

In summary, the Rössler elements are now controlled by the system defined in Eq.(7.1)

$$\begin{cases} \dot{x_i} &= \omega\big(-(y_i + z_i) + feedback_i\big) \\ \dot{y_i} &= \omega\big(x_i + ay_i\big) \\ \dot{z_i} &= \omega\big(b + z_i(x_i - c)\big) \end{cases} \tag{7.1}$$

With:

$$feedback_i = \begin{cases} K_f \cdot touch_{ipsilateral} - K_f \cdot touch_{contralateral} & \text{If } i \text{ is a leg} \\ 0 & \text{If } i \text{ is not a leg} \end{cases}$$

There exist no connection, even indirect, between different Segments. This is something that could be added for further investigation.

We then map the Rössler elements to the robot in the same way as the Rössler CPG, look in Section 5.3.

Therefore, we have the following parameters:

Body Amplitude (open) The same parameter as the Rössler CPG, controlling the gain of the Body oscillations.

Frequency (open) Again, same as the Rössler CPG. We can control it in the same way using ω and Equation (4.3).

Sensory feedback strength (open) This controls the strength and the behavior of the Sensory feedback. We need to optimize it.

We use the following parameters for the Rössler elements: $a = 0.2$, $b = 0.2$, $c = 5.7$. This creates a chaotic signal, as discussed in the first Chapters. Note that we had several problems actually creating chaos in Webots. This was due to numerical precision and integration steps. If we use an integration step that is too big, the chaotic behavior is destroyed and a periodic solution replaces it. We had to use a very small integration step to create chaos similar to the mathematical simulations.

7.1.2 Dependance on parameters

While performing the first tests of this Simple Chaotic controller, we found out that the Body Amplitude is very important. Putting a high Body Amplitude would break every effort of the controller to move. This parameter is very sensitive.

This is due to the fact that, if the robot doesn't oscillate, the legs tends to create a forward movement. This is just a consequence that the Rössler Elements can't create a decrease of the phase (see Section 8.3). So, even if we do stupid movements with the legs, if the body stays still, the robot will tend to go forward. But keeping the robot straight is obviously a bad strategy when we are on an uneven terrain, so we enforce the choice of a non-zero Body Amplitude A_k.

Dependance on Body Amplitude and Frequency

We perform again the same kind of systematic exploration of the Body Amplitude and the Frequency as we did for the Rössler CPG controller. We run 20 experiments, all with random initial conditions for the Rössler CPG. We use the travelled distance on a flat terrain as metric. We fix the Sensory feedback strength to $K_f = 0$ and put the Rössler elements into chaotic mode. See Figure 7.1 for the results.

We see that, as discussed just before, the Body Amplitude should be kept small. In fact, the optimal speed is attained with a Body Amplitude of $A_k = 0$ and a Frequency of $F = 3Hz$. But we already discussed the fact that we want to keep some Body mobility, to be able to get out of uneven terrain more easily. But we can find another point in the neighborhood of $A_k = 0$ that still gives good results, because the performance function is quite smooth. The Frequency is important too, and closely related to the movement speed. We find again a nearly linear relation between the speed, the Frequency and the Body Amplitude.

For further simulations, we choose a Body Amplitude of $A_k = 0.4$, to keep a balance between body mobility and performance. The Frequency will be kept to $F = 2Hz$, as it is our common benchmarking frequency.

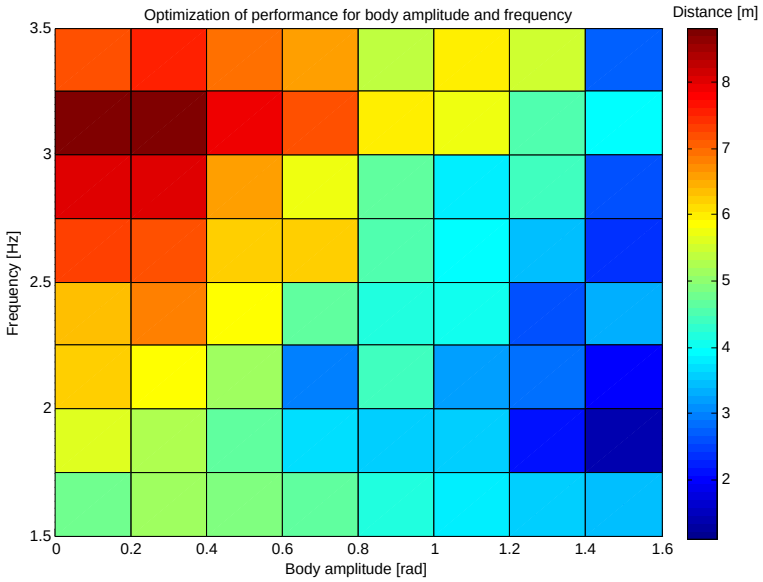


Figure 7.1: Relation between Body Amplitude and Frequency for the Chaotic controller

Optimization of sensory feedback strength

We performed a systematic search of the effect of the Sensory Feedback strength on a Simple Chaotic controller. We run 20 experiments, all with random initial conditions for the Rössler CPG, as usual. All these runs are done on a flat terrain, and we use the travelled distance as metric. We use the classical parameters of the Rössler CPG in chaotic mode, we fix the Body Amplitude to $A_k = 0.4$ and the Frequency to $F = 2Hz$. See Figure 7.2 for the results.

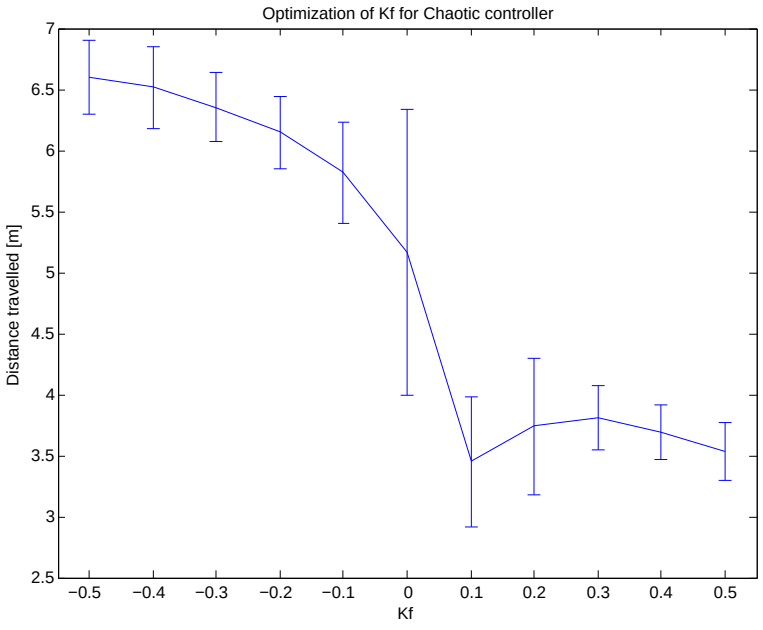


Figure 7.2: Effect of Sensory Feedback strength on performance of Chaotic controller on flat terrain.

We see that the two possibilities to include the Sensory Feedback, represented by the sign of the strength K_f differ completely in their effect. For positive K_f, we actually disrupt the movement pattern

created without feedback. This is therefore a bad choice of parameter.

For negative Sensory Feedback strength K_f, we increase the performance significantly, by around 29% with $K_f = -0.5$. So this behavior of feedback works well for the controller. If we refer to Table 6.1, we are decreasing the Stance duration and increasing a little bit the Swing duration. It means that once we touch down, we accelerate the leg. It seems that what is important here is the Frequency of the legs. But the feedback from the contralateral leg, i.e. slowing down the leg when the contralateral one touches the ground, works again this acceleration.

The plot of final positions add again good insight in the actual distribution of trajectories. See Figure 7.3 for the final positions. Again the robot starts at $(0, 0)$ and should travel the further away, along the y-axis for a stable movement. We can clearly see the two different effects of the Sensory feedback depending on the sign of K_f, symmetrically around $K_f = 0$. The dispersion is very high for most of the parameters, especially along the x-axis. It means that the robot doesn't have a tendency to go straight forward, but rather deviate on the left or on the right. It's interesting to see that without feedback, we have a very high dispersion, showing that without any feedback the Chaotic controller moves fairly randomly, but with a quasi constant forward speed. A very good surprise is that good feedbacks actually promotes a quite straight movement, along the y-axis. This adds a argument to the fact that the negative sign feedback actually promotes a synchronization between legs, balancing the robot.

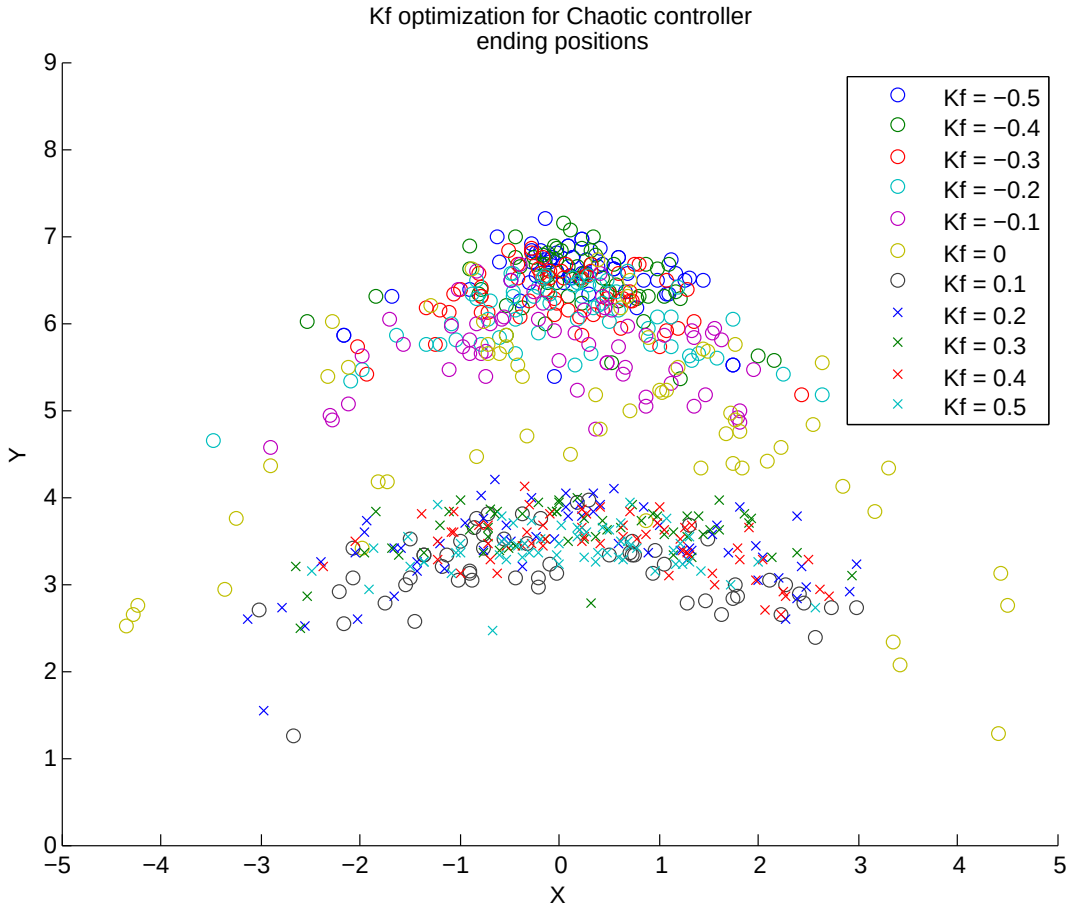


Figure 7.3: Final positions for different Sensory Feedback strength K_f for the Chaotic controller

7.2 Movement obtained and discussion

The obtained movement is less smooth than the one from the Rössler CPG in Limit Cycle mode, obviously. The Chaotic controller moves more by impulses, mainly because of the body oscillations. The legs move nearly randomly, but the Sensory Feedback tries to loosely synchronize the legs of each segment. In general the motion created is much slower than the designed locomotion pattern in Limit Cycle mode. But this is a comparison on even ground, whereas we are interested in the behavior on an uneven terrain.

A very interesting feature is the synchronization between legs created by the Sensory Feedback. The interplay between the ipsilateral and the contralateral feedback seems to create two stable phase relations between the legs:

1. They hit the ground alternatively. They thus have a phase difference close to π.
2. They hit the ground at the same time. They have a phase difference close to 0.

These two relative positions of legs were observed frequently, even if they did not stay forever. The legs would oscillate chaotically for some time, but again they synchronize again for a while to one of these two states. This is something really interesting, because this is automatically promoted by the Sensory Feedback. Before we actually hard-coded a difference of π between the legs, but now this difference arise from interactions with the environment. We will study this phenomenon in more details at the end of the benchmarking.

You can see a video of a robot with the Simple Chaotic controller working, see in the Additional Material.

7.3 Observed Chaos controller

When trying to understand the actual benefits that chaos could offer us for moving a robot, we actually came up with a new way of linking chaos with movement.

7.3.1 Theoretical concept

If we go back to a really abstract level, we can give the following statement about movement (see Eq.(7.2)):

$$movement \iff [rhythm \Rightarrow speed] \tag{7.2}$$

In other words, if we have a rhythmic process, say a circular movement, or the motion of legs, and if this rhythmic process creates a displacement, we just moved. This is very crude, but can actually be used directly as a general way of creating movement.

Imagine that we can explore the space of all rhythms. Then if we find a rhythm that creates speed, then we just need to keep doing that rhythm. If the rhythm does not produce speed, just explore again the space of all rhythms. This very general way of creating movement can be thought at every timescale, even the smallest ones. If you imagine that you are on a very hard terrain, then the rhythm that produces a movement can change completely in a very short period of time.

But then we can actually transfer this simple algorithm into a real Control methodology. Imagine that we have a source of rhythmic patterns. We just send these rhythmic patterns to the robot, and then if speed is created you continue using the same rhythmic pattern. The only problem now is how to create rhythmic patterns. We think that you can use any source of randomness able to produce rhythms usable by the robot, but the search space is obviously pretty high.

We think that Chaos could be used for this purpose. We know that Chaos can generate a lot of different frequencies. Actually one definition for chaos is that is a signal which consists of an infinity of frequencies. We could use this power to generate easily different rhythmics patterns for the robot.

But then, we add another process to this framework: an Observer. This process has a knowledge about the state of the Chaos generator, but also from the robot. The robot could send him feedback about his sensors, but also a measure a speed. We then use this Observer to force the Chaos to redo the rhythmic pattern if it did create movement. See Figure 7.4 for the block diagram of the theoretical controller.

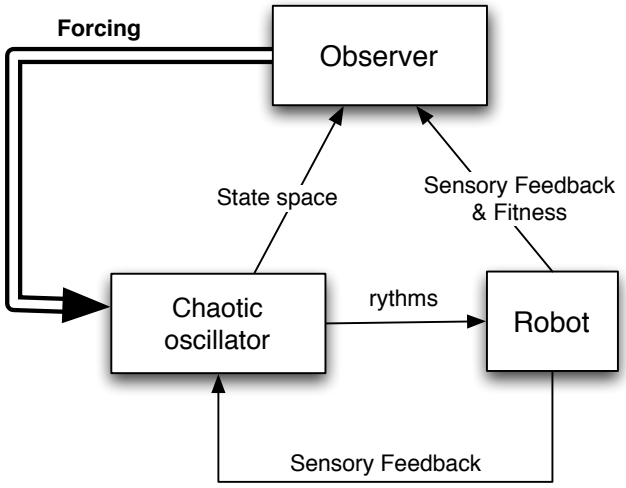


Figure 7.4: Observed Chaos controller block diagram.

The big addition here is mainly the Observer. Because before, we could actually already use the principle of generating different rhythms, but there was not any control on that. The Observer allows us to tell to the Chaos when something was good. It also allows us to keep a memory of good rhythmic patterns, and thus to actually do learning.

The good thing about using Chaos as a random generator, instead of other sources, comes from its deterministic component. If we have an autonomous dynamical system working in chaotic regime, we will always get the same output from a defined state. Even if the output seems random, it is not, and we can redo the same exact pattern just by making the chaos "start over" in a specific previous state. Note that we also connected the Robot to the Chaotic oscillators using the Sensory Feedback. This is because we think it may actually help the Chaos search the rhythmic space in a good way. This is pure speculations, but if that is the case we would have a very nice way to observe the rhythmic space.

We could also model the forcing by the Observer in a nicer way, like a coupling between dynamical systems. We could smoothly make the chaos move toward a previously good rhythmic pattern instead of just forcing it to specific state values, which seems pretty elegant.

A lot of unknowns stays with this approach. For example, is Chaos good enough at generating very different rhythms ? We saw that the Rössler system was pretty limited, so maybe it is not the best candidate. Another issue is the measure of the speed, or more generically, the definition of our metric of "good movement". If we need to change of locomotion pattern really quickly, if we are on a very hard terrain for example, then we need a metric that is a quick as possible. The Observer should also be able to determine the start and end of a rhythmic pattern, which can be tricky especially when we don't know which part of the pattern was useful (this is a credit assignment problem with agents as small as any finite interval of time).

But we think that further work in this direction could be very interesting, because it is a very general way of seeing movement generation, that could have a lot of applications.

7.3.2 Relations with other works

When seeing the block diagram of the Observed Chaos, we saw a direct relation with the work of Y. Kuniyoshi. In his work, he used chaos as a way to generate movement, which received back information from the body. By some unknown means, the chaotic elements were able to promote good locomotion patterns, just with this information.

We actually think that this can be explained using our conceptualization. If you remove the Observer in our model, then you are left with the model of Y. Kuniyoshi. In some way, they were using the same idea: let the chaos explore, and hope that it will create movement. They give sensory feedback to the chaos, hoping that it may direct it in its search of movement pattern. But the big problem with that is that we know nothing about the interplay between the sensory feedback and the chaotic generator. Moreover, the chaotic generate has no knowledge about the metric that we are trying to optimize. So by adding the Observer on top of that, we are just adding a level of control. We are adding the process that can control the behavior of the chaotic generator, to really make it do things that we decided were useful. Moreover, we have a total control on the metric we use, and on the way to direct the chaotic generator in his search of rhythmic patterns. This explains the fact that we felt like something was missing to the work of Y. Kuniyoshi, we think this Observer process is a possible way to fill the hole.

We think that another behavior of the chaotic elements of Y. Kuniyoshi fits perfectly into our framework: with his chaotic controller, he was able to promote locomotion pattern for different robots. But the locomotion patterns were not stable. They would occur for some time, but then change completely. The robot would for example change direction, go backward, stays, without any control on the different behaviors. If you think in term of generation of rhythms, this is a direct consequence of the re exploration of the rhythmic space. For some reason, the chaotic generator was locked on a specific rhythm, but then it went away from it (because of a bifurcation, or because it was not a stable attractor) and created another rhythm. It would then move from rhythm to rhythm, without any real control on that because there was no Observer to make it continue for defined period of time.

Further work on this has to be done, but we think that applying the Observed Chaos pattern could give the control over the emergence of movement that Y. Kuniyoshi managed to create.

Chapter 8

Benchmarking on uneven terrain

We now have all we need to answer the actual question that motivated this work:

- How does a Chaotic controller stand compared to a completely design Limit Cycle controller and a completely free Random controller ?

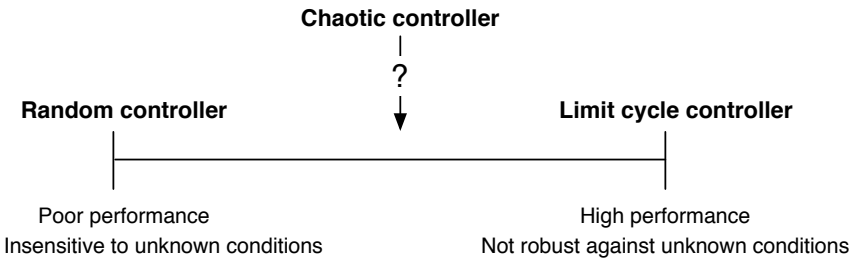


Figure 8.1: Is a Chaotic controller standing between a Random controller and a Limit cycle controller ?

See Figure 8.1 for a graphical interpretation. We make here the assumption that there are two "extrema" of controllers:

1. You can build a controller in order to execute a precise desired movement pattern. Usually, you want this movement pattern to be effective, very well adapted to your robot and therefore easy to predict. You can get high performance using this approach, but as soon as the environment differs from what you assumed when building the movement pattern, the performances are destroyed.

2. The controller is not designed at all, it is doing something totally random. Or the design is very crude and lets a lot of random do the work instead. Obviously you will not get high performances out of this. But you don't need assumptions on the environment. It corresponds to some kind of minimal bound of performance for a controller.

Our hypothesis is that the Chaotic controller lies somewhere between those two extrema: it is not very designed, and because of its chaoticity behave nearly randomly, but by some kind of interplay, a design emerges automatically. We want to assess experimentally this hypothesis, and we will develop a test bench for that.

8.1 Goal and description

As our assumption on the Limit Cycle controller states that the controller drops in performance when the environment becomes unknown or difficult, we will systematically quantify this drop by creating

different environments of growing difficulty. We create an environment which consists of a Test area of a defined difficulty. This difficulty corresponds to the degree of variation of the ground. Consider for example a terrain with holes or bumps. Its difficulty could be the density or height of such holes and bumps. We want to see how the controllers behave on these different terrains.

The goal for the controllers is to move as far as possible, while being on this Test area.

8.2 Random uneven terrain generator

We chose to generate these uneven terrain randomly, to avoid as much as possible undesirables side-effect of manual engineering of terrains (e.g. symmetries). We developed a program to create automatically Webots' worlds containing a random Test area, called *tergen*. It is available in the Additional Material. See Figure 8.2 for a schema of the worlds we are creating.

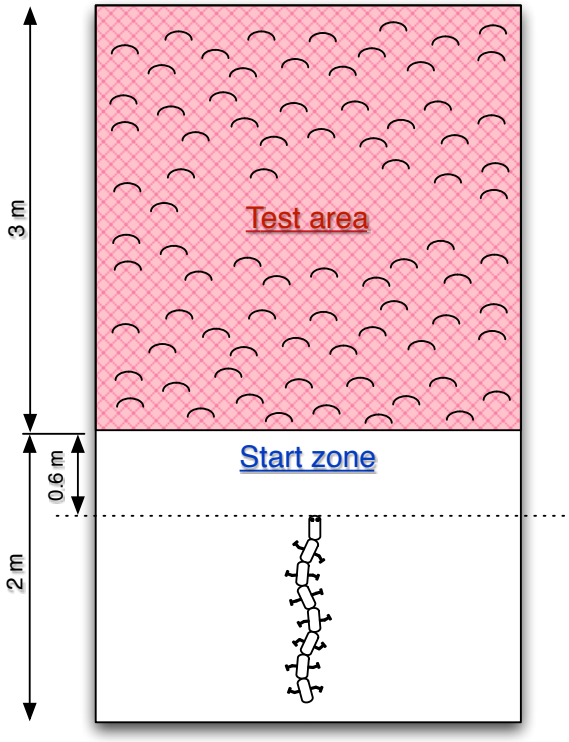


Figure 8.2: Experiment area schema.

The Test area can be defined in any way. We chose to create a mesh of triangles with random heights. The difficulty is defined as the percentage of the maximum derivative possible between two adjacent vertices :

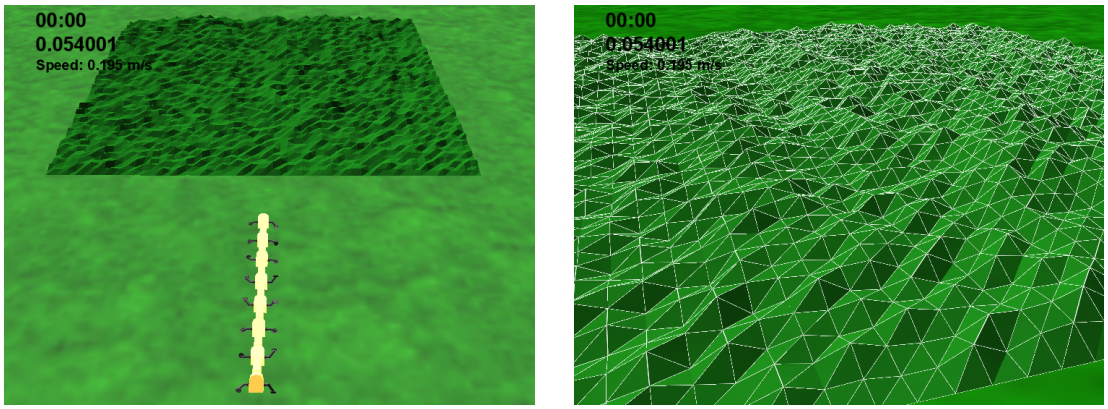
$$z_i = rnd() \cdot height_{max}$$

with:

$$|z_i - z_k| < difficulty \cdot derivate_{max} \quad \forall k \text{ adjacent to } i$$

We have 10 different degree of difficulty, $difficulty \in [0.1, 1.0]$. A set of world consists of 10 worlds of all possible difficulties. The terrain created model accurately growing levels of difficulties for the movement of a robot. But we need to precise that most of the terrain with difficulties higher than 0.5 are far too complicated for any real robot to move upon. Here we are working with a physical simulator

and a model of a robot. This model is quite permissive, we could not do the same in reality. But the principle behind it is correct and applicable in real world too. See Figure 8.3 for an example of a mid-level difficulty terrain (0.5).



(a) General view of the Experiment area.

(b) Close view of the uneven terrain.

Figure 8.3: View of the Webots simulation of the Experiment area. World of difficulty 0.5.

8.3 Building a fair random controller

In order to find out if a Chaotic controller is more than just a random controller, we need to build a Random controller. There are several ways of constructing a Random controller, especially knowing the number of degrees of freedom we have on our robot.

But the main problem was to build a "fair" random controller. That is, build a random controller that has the same capabilities than the Limit cycle Controller and the Chaotic controller. There is indeed a major advantage given to those two controllers:

- The legs never go backward.

This is due to the behavior of the Rössler system, and the way we defined the phase. The trajectory is indeed always turning in the same direction (to its left), because the chaoticity only introduce jumps out of the XY-plan, without breaking the cyclic behavior of the attractor. The Rössler system is too simplistic in this way, it produces a very simple behavior.

This means that, if we let our robot straight (no body oscillations) and let the legs oscillators move, even with very bad parameter settings, the robot will tend to go forward. The legs will always turn in the right direction and push the robot forward.

So we need to take that into account when building a Random controller, because we want to have a fair comparison.

8.4 Forward random controller

Since our robot takes only cyclic inputs, it is plausible to create signals out of a circular motion. That is, we simply go along a circle, but the speed we go along it is random. See Figure 8.4 for an illustration. This is simply modeled as a phase Θ_k variable.

As we want to create a random movement that has the same advantage as the Rössler elements, we only allow the movement in on direction along the circle. That is, the phase Θ_k can only be increased.

We update Θ_k in the following way:

$$\frac{d\Theta_k}{dt} = 2\pi \cdot \omega \cdot U(0,1) \tag{8.1}$$

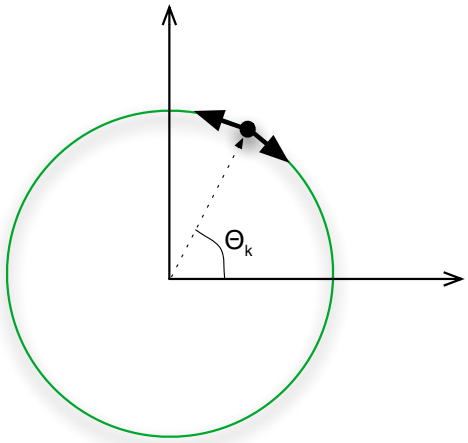


Figure 8.4: Random movement along a circle

Where:

- $U(0,1)$ is a random variable following an uniform distribution between 0 and 1.
- ω is a parameter controlling the average frequency of movement.

We performed several tests and analyzed the frequencies using a FFT as done in Section 4.1.3. We found out the following relation between ω and F_{elem}, the average frequency of one of the Θ_k variables, see Eq. (8.2).

$$F_{elem} = \frac{w}{2} \tag{8.2}$$

We simply integrate just like before these new Θ_k variables.
Finally, the mapping to the robot's input is done in the following way:

Legs Use the phases directly, this is exactly the input of a leg. $y_{leg_k} = \Theta_k$.

Body segment Put the phases in a sinus and control its amplitude, just like in the case of Rössler CPG, or the previous work of B. Jimenez. The mapping function between the state and the robot output is then: $y_{segment_k} = A_k \sin(\Theta_k)$.

The obtained controller has a tendency to go forward, at a speed controlled by the parameter ω.

8.5 Pure Random controller

To show if the Rössler really had an intrinsic advantage, we also build a controller which doesn't have the tendency to go forward.

We use the same idea as before, for the Forward Random, but we change the update rule for Θ_k in the following manner:

$$\frac{d\Theta_k}{dt} = 2\pi \cdot \omega \cdot (2 \cdot U(0,1) - 1) \tag{8.3}$$

The random component is now symmetric around 0, which allows backward movements too. The ω parameter doesn't control the frequency in the same way as before. The symmetry induce a mean frequency of 0, so the only way to make it move is to put a big ω, which will act like a gain. Values around $\omega = 15$ give satisfactory movements of legs and body.

Trivially, this controller has very little chances of producing a good movement. More precisely it has very little chance to produce any movement at all. But we want to know where the Chaotic controller is located among possible controllers.

8.6 Benchmarking parameters

We test every controller with different parameters sets, on every terrain, for a given experiment time of 20 seconds. This time has been chosen because it was the time needed to go across a Test area of difficulty 1 using a Limit cycle controller.

We perform 120 runs for each of the controllers and each of the worlds. For each run, we choose random initial conditions for the controllers. Moreover, we generate 8 different random sets of worlds, on which we will carry the experiments. Generating multiple random worlds ensures to neglect topological problems, and helps to actually observe the behavior of the controllers with respect to difficulty levels.

We will perform those tests at two different frequencies, $F = 2Hz$ and $F = 3Hz$. The frequency has a major role in the performance, so we chose not to mix results from different frequencies.

Here are the different controllers tested and their parameter sets (Table 8.1 and Table 8.2):

Limit Cycle We use the three parameter sets defined in Table 5.1.

Random Forward and Pure We choose a Body Amplitude of $A_k = 0.4$, to fit with the Chaotic controller amplitude. The ω parameter is chosen to produce either $F = 2Hz$ or $F = 3Hz$

Chaos Optimized This has been chosen to reflect the optimizations done before, while still using a Body amplitude bigger than 0 to keep some mobility on the high difficulties worlds. This was an a-priori choice, we confirm it by doing a systematic search on the Body Amplitude on every difficulty level of worlds. We use the Chaos Optimized controller for that, and perform 60 runs for each world and each body amplitude, see Figure 8.5. It confirms our hypothesis, in high difficulty worlds an Amplitude of $A_k = 0.3$ is the optimal choice, while keeping good results on flat worlds.

Chaos Bad We also use a Chaotic controller without feedback to see its effect in the emergence of coherent behavior.

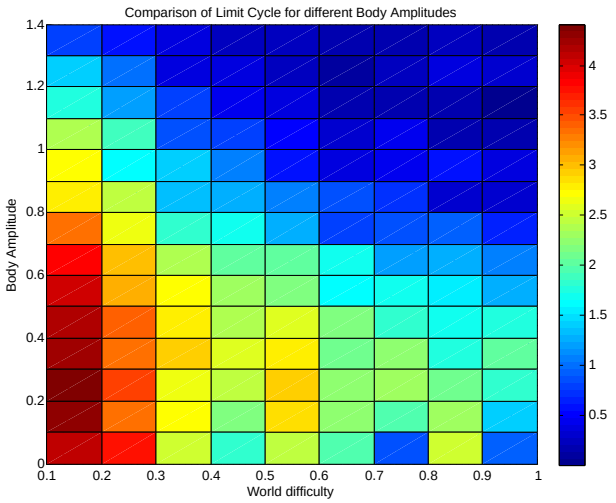


Figure 8.5: Optimization of Body Amplitude for Chaotic controller on different worlds

	Optimized Limit Cycle	**Bad Limit Cycle**	**Bad Limit Cycle Sensory**
Body ampl. A_k	1.6	0.8	0.8
ω	6π	6π	6π
Bias leg $bias_{leg}$	2.93	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
Sensory feedback K_f	0	0	0.5
	Random Forward	**Random Pure**	
Body ampl. A_k	0.4	0.4	
ω	4.0	4.0	
	Chaos Bad	**Chaos Optimized**	
Body ampl. A_k	0.3	0.3	
ω	6π	6π	
Sensory feedback K_f	0.0	-0.5	

Table 8.1: Parameter sets for frequency $F = 2Hz$

	Optimized Limit Cycle	**Bad Limit Cycle**	**Bad Limit Cycle Sensory**
Body ampl. A_k	1.4	0.8	0.8
ω	9π	9π	9π
Bias leg $bias_{leg}$	2.93	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
Sensory feedback K_f	0	0	0.5
	Random Forward	**Random Pure**	
Body ampl. A_k	0.4	0.4	
ω	6.0	6.0	
	Chaos Bad	**Chaos Optimized**	
Body ampl. A_k	0.3	0.3	
ω	9π	9π	
Sensory feedback K_f	0.0	-0.5	

Table 8.2: Parameter sets for frequency $F = 3Hz$

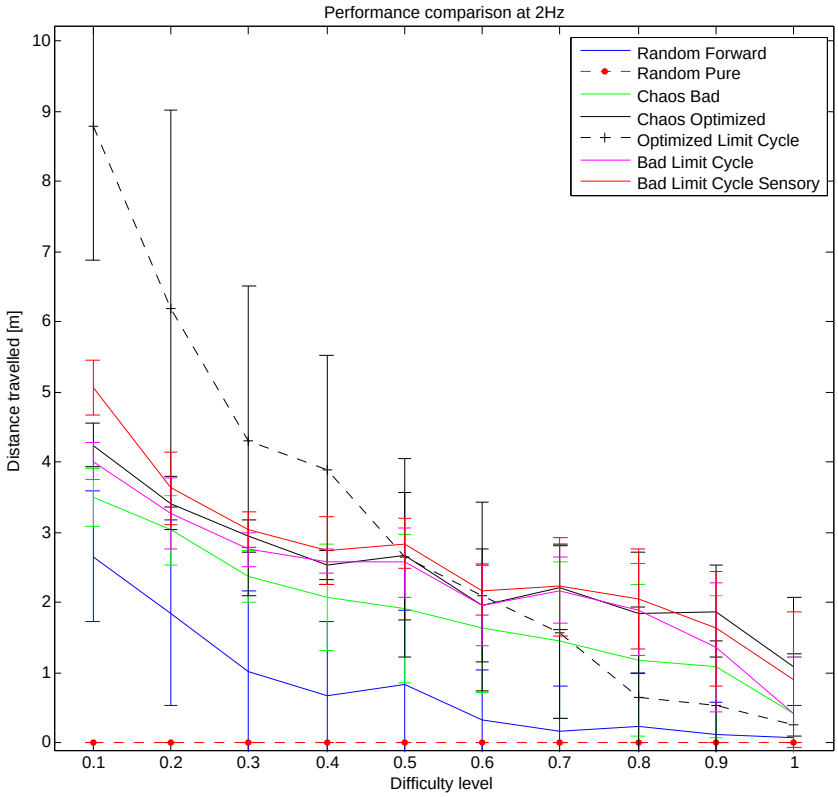


Figure 8.6: Benchmark results for all controllers with $F = 2Hz$. Distance travelled on Test Area on the y-axis, different difficulty levels on the x-axis.

8.7 Benchmark result for Frequency $F = 2Hz$

See the results on Figure 8.6 for every parameter set and every world. We can observe several phenomena happening, we are going to present them precisely.

8.7.1 Performance of Limit Cycle

As expected, the Optimized Limit Cycle is the best performer when the terrain is easy. It is nearly 2 times more successful than its competitors. But we see, as hypothesized, that its performances drop dramatically as soon as the difficulty increases. It nearly doesn't move when the difficulty is maximal, which is very extreme.

On the other hand, the Bad Limit Cycle behave poorly on easy terrain, being sub-optimal, but stays quite good when the difficulty increases. It is only when the difficulty grows above 8 that its performances become really bad. Then at difficulty 10 it does not move anymore, like the Optimized Limit Cycle. We think that this difference for higher difficulties comes from the Body Amplitude. The Bad Limit Cycle has a smaller Body amplitude, and it seems that this is a very important parameter for high difficulty worlds. A robot with a high amplitude has more chances to get stuck in small holes, and this seems to decrease a lot the performances.

We see something very interesting with the Bad Limit Cycle Sensory: It is always better than the Bad Limit Cycle. We can also see that the increase in performance is nearly constant along difficulty levels. The Sensory feedback thus seems to help moving the robot, even when the terrain is uneven. It is even more useful when the terrain is really difficult (difficulty 10): in that case, it allows the robot to move despite the terrain, which wasn't possible before. It means that our Sensory feedback indeed help when the terrain is difficult, which is a good thing because we did not design it specially for that

purpose.

8.7.2 Random controllers

The absolute non-movement of the Random Pure controller confirms what we thought: The Rössler CPG has an intrinsic advantage over pure random movements. Using pure random movement does not produce any coherent behavior.

On the other hand, the Random Forward is performing quite well. We see that on a easy terrain, it actually moves quite nicely. It stays the least effective in its task, but we think this is coherent with the assumptions we made. We also see that the Random Forward controller sees its performance decrease when the difficulty increases. This was not expected, as we thought that random movement should not be influenced by the difficulty of the terrain. But it is the case, which shows again that the other controllers do a good job at moving correctly on these terrains. We think again that the Body amplitude could play a role here, maybe it is still too big on the high difficulty terrains.

We tried to add Sensory feedback to the Random Forward controller, in a similar way than for the Rössler CPG. We thought it may improve the performance of the Random Forward to something more closer to the Chaotic controllers. But we were wrong, there was absolutely no significative difference between the Random with or without Feedback. Therefore we don't show it on the figure. We think it may be due to the way we incorporate the Sensory feedback, maybe it was not working the same than for the Rössler CPG.

8.7.3 Chaotic controllers

We get at least to the results that we were searching for all this project. They are quite surprising in fact: the chaotic controllers behave better than we would have thought.

The Chaos Bad, which is just a chaotic controller without any feedback and links between its elements, performs quite well. It actually outperforms the Random Forward by a large amount, despite the fact that they have a very similar pattern. If the chaos is used here as a source of unknown noise, it seems that this noise is pretty effective for making forward movements. Moreover, the Chaos Bad keeps good performances even when the difficulty increases. This controller corresponds to the random controller we made the assumption upon in the beginning of the Chapter, in the contrary of the designed Random Forward. But we see also that, for the highest difficulty, the Chaos Bad doesn't move at all, just like the previous controllers without Feedback.

The Chaos Optimized is really surprising. It manages to get performances above the Bad Limit Cycle, even on flat terrain. It lays between the Bad Limit Cycle and the Bad Limit Cycle Sensory. Remember that we did not design any motion pattern for this controller, we just put the Sensory Feedback. Even so, the controller manages a fairly good performance, which becomes pretty amazing for high difficulty worlds.

For a difficulty of 0.5, the Optimized Limit Cycle gets pretty bad results, and actually does the same performance as the Chaotic Optimized and the two other Limit Cycle. The Optimized Limit Cycle continues to drop for higher difficulties, while the Chaotic Optimized stays fairly constant. This is exactly what we wanted to assess: The Chaos Optimized performs better than a optimized movement pattern in unknown terrain, and at the same time is better than just random.

The Chaos Optimized continues to stay quite constant for difficulties of 0.9 and 1.0, and in fact actually gets higher performances than all other controllers for those two really difficult terrains. This is way beyond our expectations, so we really want to assess this theoretically.

8.7.4 Kruskal-Wallis tests

To statistically assess the performances of the controllers, we perform a non-parametric Kruskal-Wallis test on the results for the terrains of difficulty 0.1 and 1.0. See Figure 8.7 for a recapitulation of the performances of different controllers on the easiest world, and Figure 8.8 for the same on the hardest world.

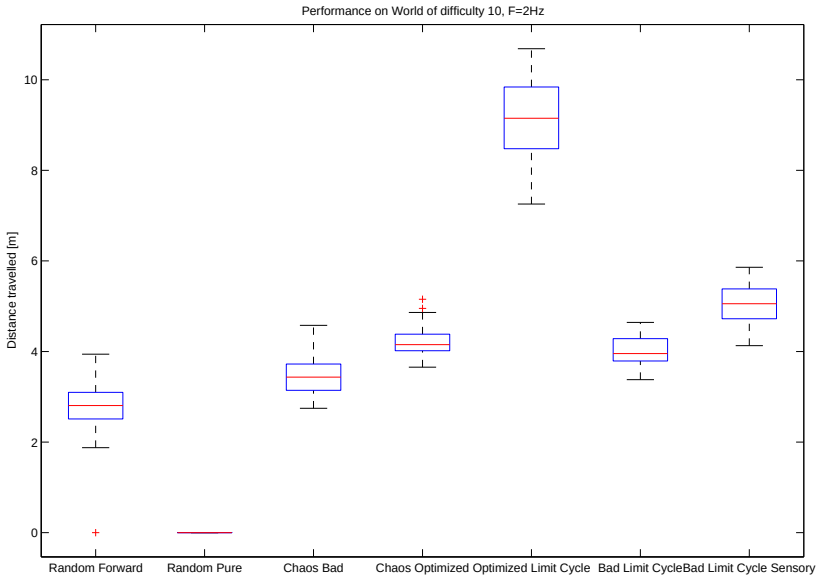


Figure 8.7: Performances of controllers on terrain of difficulty 0.1.

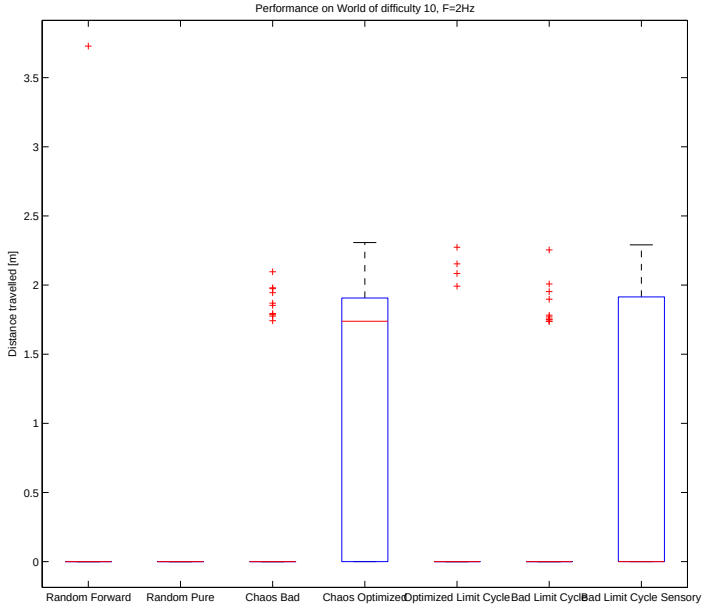


Figure 8.8: Performances of controllers on terrain of difficulty 1.0.

We then perform a Kruskal-Wallis test, for individual independence of the controllers. The test is done for a 95% confidence interval. The results are shown on Figure 8.9 for the world of difficulty 1.

We see that the controllers can be distinguished, at the exception of the Chaos Optimized and the Bad Limit Cycle. What interests us the independance of the Chaos Optimized with respect to the Optimized Limit Cycle and the Random Forward. We are clearly performing between these two controllers on an

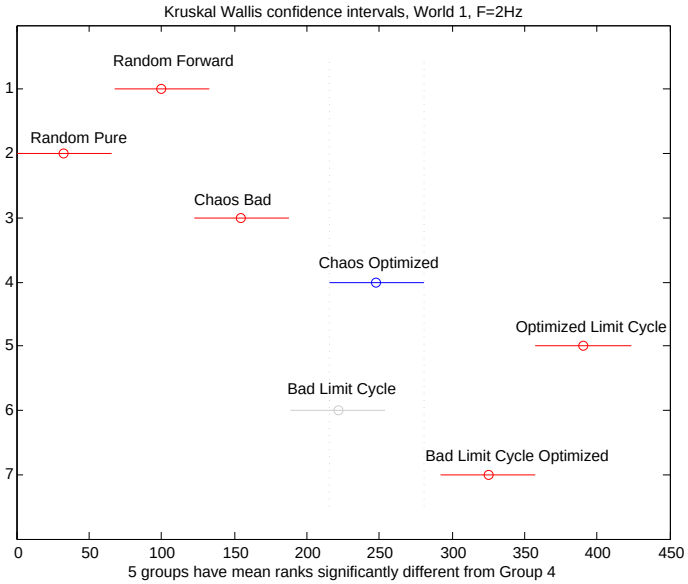


Figure 8.9: Kruskal-Wallis confidence intervals for World 1. Intervals that don't overlap are independant with 95% confidence.

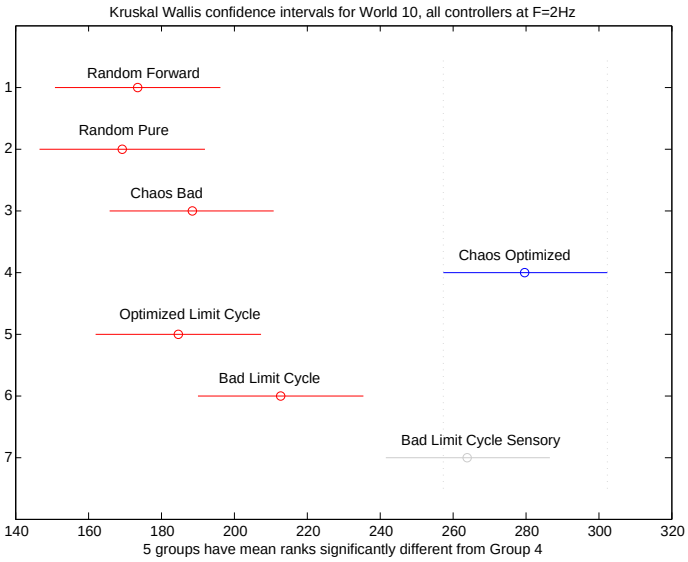


Figure 8.10: Kruskal-Wallis confidence intervals for World 10. Intervals that don't overlap are independant with 95% confidence.

easy terrain.

Then see Figure 8.10 for the world of difficulty 10.

Again here there is a significative difference between the Chaos Optimized and the other controllers. We see that almost all the controllers behave really bad. In fact, only the two controllers which have Sensory Feedback on actually get a significative performance. This is really interesting, and again shows that our Sensory feedback is useful.

For the Chaos Optimized, we see that it is statistically better than the Optimized Limit Cycle and the Random Forward. This is a really good thing, and wasn't assessed before doing this experiment. What is also interesting is the fact that the Sensory Feedback is needed for the Chaotic controller to perform well. The good performance of the Chaotic controller comes then from the added information

of the feedback, and most likely from the indirect coupling that it offers.

We performed a systematic search on the effect of different Sensory Feedback strength for all Worlds, to check if the optimization we did before is still valuable with uneven terrain. See Figure 8.11 for the results. We see that the landscape is indeed the same, and that a negative Sensory feedback is the good choice for the Chaotic controller.

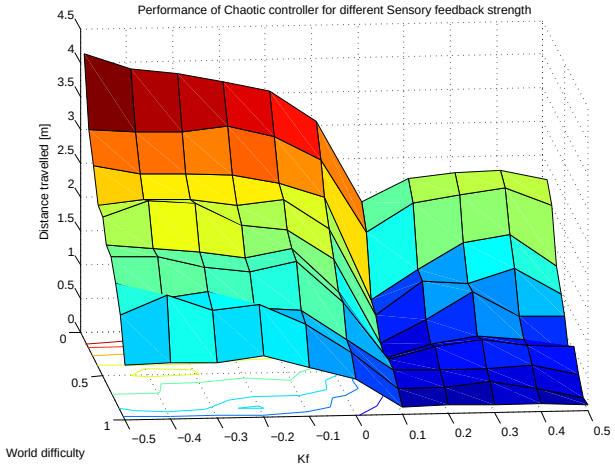


Figure 8.11: Performance of Chaotic controller depending on the Sensory feedback strength

8.8 Benchmark results for Frequency $F = 3Hz$

In order to confirm those results, we performed the same experiment but with a Frequency of $F = 3Hz$. By doing this we avoid the situation where the outcome was created by rhythmic relations between the robots and the controller, without real explanations from the controller behavior.

See Figure 8.12 for the results at $F = 3Hz$.

Most of the observations are still the same at 3Hz, but let's see the differences:

- The Random Forward is working better at that Frequency. It is nearly equal to the Bad Limit Cycles, and we can't distinguish it from Chaos Optimized up to World 0.5. It keeps a fairly constant performance independently of the difficulty. This looks much more like what we assumed about a random controller. This is most likely due to the higher frequency, which decreases the Body amplitude obtained, and promotes the forward movement easier.

- The Optimized Limit Cycle drops in performance, but stays good a little longer. This is both because its Amplitude is smaller than at $3Hz$ and because its movement pattern allows it to move really quickly if it manages to move on the uneven terrain.

- The Bad Limit Cycle does not really profit from the Sensory feedback now. We think this is because the speed of the legs is too big to allow the Feedback to influence the controller effectively. Again its seems that the small body Amplitude is better to move on uneven terrain compared to the Optimized Limit Cycle.

- The Chaos Optimized is indistinguishable from the Random Forward at first, which is a bad thing because we exactly want to prove the inverse. But this state of fact is only true for easy worlds, and when the terrain become more complicated, the Chaos Optimized starts to stand out. It again becomes better than the Optimized Limit Cycle, but now at the difficulty level 0.7. Again the Chaos Optimized is the better controller on the hardest worlds, which confirm the observations of $F = 2Hz$.

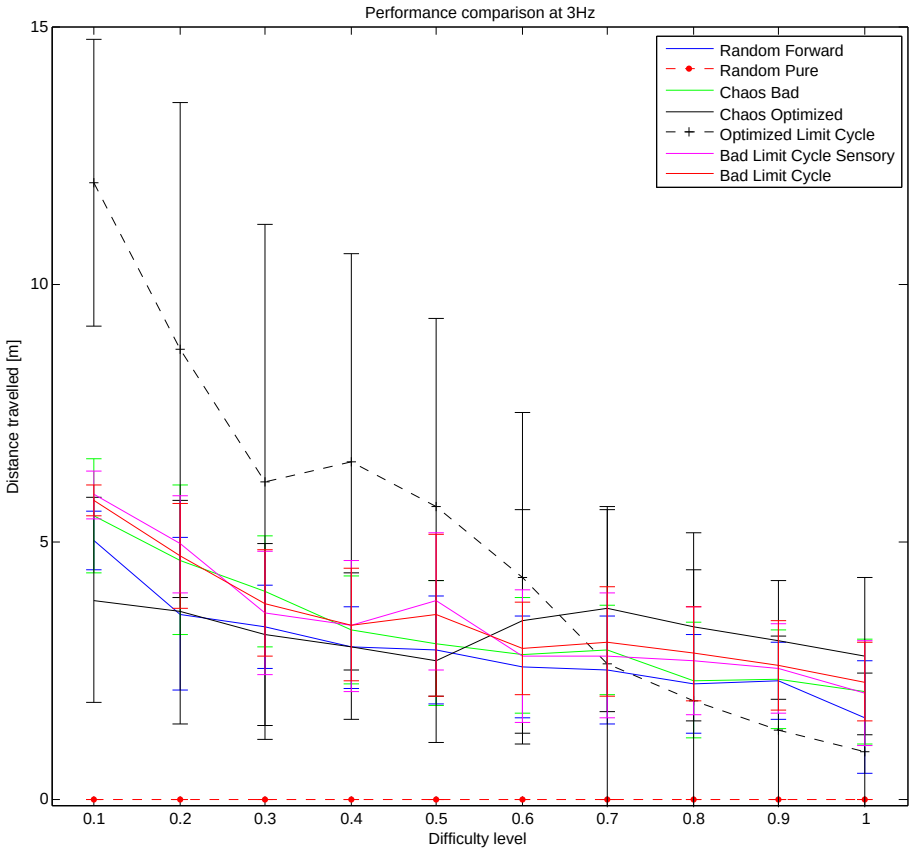


Figure 8.12: Performance comparison, at $F = 3Hz$

We perform statistical Kruskal-Wallis tests again, on the World 10 results. See Figure 8.13 for the recapitulation of the performances on this world, and Figure 8.14 for the confidences intervals. Again these are 95% confidence intervals, if they don't overlap then the controllers that created them are independent.

The results from the Kruskal-Wallis test are less definitive than for $F = 2Hz$. For this higher frequency, the Chaos Optimized is not statistically different than Chaos Pure or Bad Limit Cycles. But you see that there seems to be an effect for the Sensory Feedback, even if it is not that visible. What still stays very good is the difference between the Chaos Optimized, the Random Forward and the Optimized Limit Cycle. They are independent, meaning that the Chaos Optimized is really better than both Random Forward and Optimized Limit Cycle.

This confirms then the results from frequency $F = 2Hz$. We can therefore assume that the Chaos brings something that does not simplifies itself to a Random controller.

8.9 Is Chaos more than just a Random movement ?

We arrive to the final point of this work, we are now in position to answer to the question that motivated this work.

From the systematic tests that we performed, and considering their statistical relevance, but also from a visual point of view of the movement pattern created by the Chaos Optimized controller, we can say that: **Yes**, Chaos is more than just a Random movement.

The overall performance of the Chaotic controller lies between a random controller and a fully designed locomotion pattern, the Optimized Limit Cycle in our case. Moreover, it is one of the most performing controller on uneven terrain, which is also a good point for its usefulness.

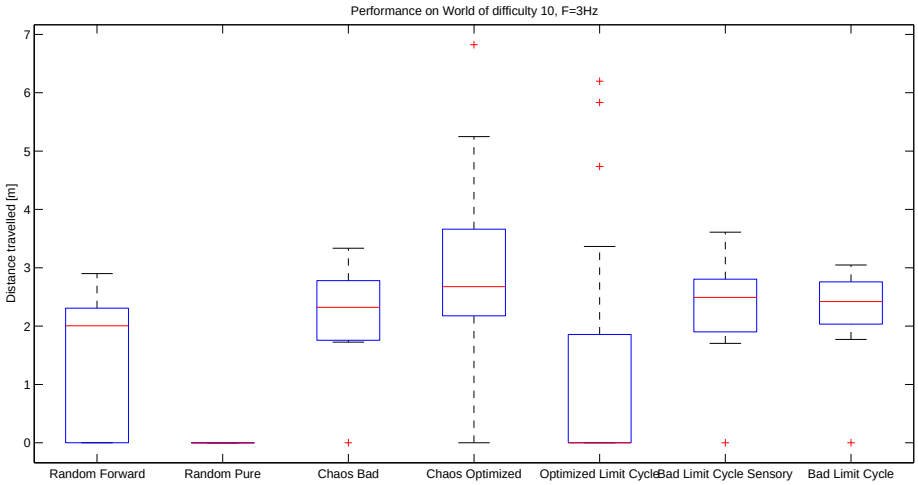


Figure 8.13: Performances of all controllers on World 10, at Frequency $F = 3Hz$

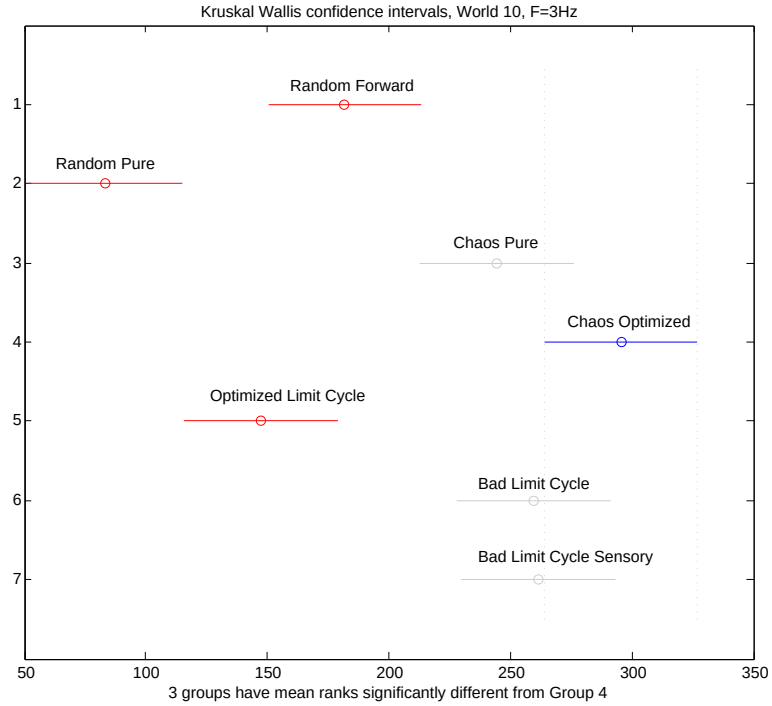


Figure 8.14: Kruskal-Wallis confidence intervals for World 10, at Frequency $F = 3Hz$. Intervals that don't overlap are independant with 95% confidence.

But now we still have a problem: we can't give real theoretical insight in the reasons why this is so. We have several ideas:

- The Sensory Feedback adapts the frequency to the body, and increases it a little bit when it is small.
- The Body amplitude is kept small because of the chaotic dynamics, with short bursts of high amplitudes from time to time. This allows the robot to get out of bad terrain, while still allowing

a movement that tends to go forwards.

- The Sensory Feedback we used tends to create two stable legs movement patterns: in-phase synchronized or anti-phase synchronized. These two kind of movement patterns are often distributed over the whole body, with for example alternations between in-phase and anti-phase legs. The anti-phase pattern creates a constant forward movement, at a low rate but that works well on flat terrain. The in-phase pattern creates bursts of movements. When the two legs touch the ground, it is likely to make the robot move, even if the terrain is very difficult. But this is not optimal on flat ground between the time between two pushes is high. While alternating these two patterns, the Chaos Optimized controller manages to keep a good movement speed, independently of the terrain. This is not yet understood why the Sensory Feedback we used promotes those two patterns, and if they are stable for long periods of time.

Chapter 9

Conclusion and Outlook

9.1 Conclusion

During this project, we wanted to assess the relevance of using Chaos to control a Centipede robot.

We started by developing a new CPG, composed of dynamical elements that could bifurcate between a limit cycle behavior or chaotic behavior depending on a parameter value. We chose to use a Rössler oscillator as our basic element. We studied its capabilities and explored its parameters range. We then had to develop a coupling scheme that would allow us to create arbitrary phase locking relations between elements. We successfully adapted Diffusive coupling, by rotation terms for this purpose. Our coupling scheme allowed to define arbitrary phase synchronization, as well as complete synchronization. Complete synchronization was also possible while the element was in chaotic regime. Phase synchronization in chaotic regime is not possible for every phase, but only for values from 0 to 0.9 radians. Afterwards, we designed a locomotion pattern adapted to a Centipede robot, built on the previous work of B. Jimenez [20]. We created this locomotion pattern with our Rössler CPG working in Limit cycle mode. Its mathematical stability was assessed.

We implemented this Rössler CPG on a controller for the Centipede robot, and thoroughly tested the parameters controlling its movement. We optimized most of them, and managed to create a controller that performed an optimal movement pattern (to the best of our knowledge). We then added Sensory feedback on this robot model and tried several ways to use it into our controller. We then optimized its coupling scheme. We found out that the Sensory feedback is not needed on flat terrain for an optimal controller, but it did increase significantly the performance of a non-optimal controller, by around 30%.

After that, we modified our Rössler CPG to make it work in Chaotic regime. We remove all the design we used in Limit Cycle mode, and every Rössler element is now independent. We incorporate Sensory Feedback to these elements, hypothesizing that it is the feedback that actually creates the relations between the elements. We think these links would emerge from interactions with the outside world. We then optimized the feedback strength and its direction. We also introduced a new theoretical conceptualization for the use of Chaos in a controller. This conceptualization can actually help to explain why the feedback is needed and how it can help to promote good locomotion patterns.

Finally, we developed a whole Test bench to systematically assess the performance of the different controllers on unknown terrain. These terrains were parametrized by a difficulty level. We also developed a fair Random controller, to see where the Chaotic controller would fit in between a completely random controller and a completely designed controller. Our results were surprising, and confirmed the fact that the Chaotic controller indeed performed well. The Chaotic controller performed best when acting on difficult terrain. This observation was verified using non-parametric Kruskal-Wallis variance tests. The Chaotic controller got a medium performance on flat terrain, where a totally designed controller

performed best. But as soon as the difficulty increased, the designed controller dropped dramatically in performance, while the Chaotic controller stayed good.

This bring us a direct use of our Rössler CPG to build a new controller: Combine the two behavior and switch between Limit Cycle mode and Chaotic mode depending on the actual terrain type. Such a change is really easy to do and completely built-in in our controller. We also need to define a way to detect the difficulty of the current terrain. Then we would get a controller that is able to move optimally on flat terrain, and could change to a very robust chaotic controller on uneven terrain. Such a "Hybrid controller" could be very interesting to study.

9.2 Outlook

Further work may include the following ideas:

1. We could develop this "Hybrid controller" and assess its capabilities experimentally.
2. The Sensory feedback that we use doesn't project onto the Body segments, we could try to add this.
3. For now, the real relation between the Sensory Feedback and the Chaotic behavior on a mathematical level is completely unknown. Better mathematical insight in the synchronization and coupling processes taking place could be very useful to try to answer to the real question that is still completely unknown: Why does the chaos perform that well ? If we think that the chaos is just used as a source of noise, we could try to recreate the exact same noise, and then see if we obtain the same behavior. If we don't, it means that the dynamical system does something more and actually uses the feedback in a way that promotes the movement.

There is still a lot of work to be done, we just did the first steps, which confirmed that Chaos is interesting to control robots. This is a very interesting answer, because there is for now nearly nobody trying to use Chaos for Robot Control. If we manage to understand the processes taking place between the Feedback and the Chaos dynamics, we could actually use this power as a new way to solve Control problems. We hope that this project opened a door in this direction.

Chapter 10

Bibliography and Additional Material

10.1 Videos

rossler_lc_bad.mpeg Shows the movement obtained with the Bad Limit Cycle parameter set for the Rössler CPG in Limit cycle mode.

rossler_lc_optimized.mpeg Shows the movement obtained with the Optimized Limit Cycle parameter set for the Rössler CPG in Limit cycle mode.

chaotic_controller.mpeg Shows the movement obtained with the Simple Chaotic controller.

rossler_lc_uneven_diff1.mpeg Shows the movement of the Optimized Limit Cycle on uneven terrain of difficulty 1. Difficulties 3, 5 and 10 are also shown on similar videos.

10.2 Acknowledgement

I would like to especially thanks and acknowledge:
Ludovic Righetti for all help, his understanding and the great discussions about obscure mathematical notions.
Prof. Auke Jan Ijspeert for his simple advices that always solve complex problems.
Yvan Bourquin and Alessandro Crespi for all the technical support and the room full of unused computers.
and all the crazy people working in the students' Lab for making every day of work so special.

Bibliography

[1] B. Anderson, J. Shultz, and B. Jayne. Axial kinematics and muscle activity during terrestrial locomotion of the centipede scolopendra heros. J Exp Biol, 198(Pt 5):1185–95, Jan 1995.

[2] B. R. Andrievskii and A. L. Fradkov. Control of chaos: Methods and applications. i. methods, Jan 2003.

[3] P. Arena, L. Fortuna, M. Frasca, G. L. Turco, L. Patané, and R. Russo. Perception-based navigation through weak chaos control. 2005.

[4] M. Biehl. Semester project : Study of coupled chaotic systems for dynamic emergence of locomotion. Master's thesis, Jul 2006.

[5] S. Boccaletti, J. Kurths, G. Osipov, D. L. Valladares, and C. Zhou. The synchronization of chaotic systems. Physics Reports, 366:1–2, 2002.

[6] V. E. Bondarenko. 'anticontrol' of chaos in an analog neural network by external low-dimensional chaotic force. 1998.

[7] J. Buchli, F. Iida, and A. J. Ijspeert. Finding resonance: Adaptive frequency oscillators for dynamic legged locomotion. Proceedings of the 2006 IEEE/RSJ international conference on ..., Jan 2006.

[8] J. Buchli, L. Righetti, and A. J. Ijspeert. Adaptive frequency oscillators applied to dynamic walking: Ii. adapting to resonant body dynamics. Proceedings of Dynamic Walking 2006 (this volume), Jan 2006.

[9] G. Chen and X. Yu. On time-delayed feedback control of chaotic systems. Circuits and Systems I: Fundamental Theory and Applications, Jan 1999.

[10] B. Duran, G. Metta, and G. Sandini. Emergence of smooth pursuit using chaos. 2007.

[11] France5. France 5 education.

[12] J. Freeman. Mass action in the nervous systemexamination of the neurophysiological basis ... Book, 2004.

[13] R. Ghanea-Hercock and D. P. Barnes. Mobile robot dynamics: Chaos in reactive control architectures. 1995.

[14] J. M. Hausdorff, C. K. Peng, Z. Ladin, J. Y. Wei, and A. L. Goldberger. Is walking a random walk? evidence for long-range correlations in stride interval of human gait. J Appl Physiol, 78(1):349–58, Jan 1995.

[15] A. Hramov and A. Koronovskii. An approach to chaotic synchronization. Arxiv preprint nlin.CD, Jan 2005.

[16] A. Ijspeert. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated .... Biological Cybernetics, Jan 2001.

[17] A. J. Ijspeert. Dynamical principles in neuronal systems and robotics. Biological cybernetics, 95(6):517–8, Dec 2006.

[18] A. J. Ijspeert, A. Crespi, and J.-M. Cabelguen. Simulation and robotics studies of salamander locomotion: Applying neurobiological principles to the .... Neuroinformatics, Jan 2005.

[19] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. Science, 315(5817):1416–20, Mar 2007.

[20] B. Jimenez. Centipede robot locomotion. 2007.

[21] Y. Kuniyoshi and S. Sangawa. Early motor development from partially ordered neural-body dynamics: experiments with a cortico- .... Biological cybernetics, Jan 2006.

[22] Y. Kuniyoshi and S. Suzuki. Dynamic emergence and adaptation of behavior through embodiment as coupled chaotic field. Intelligent Robots and Systems, 2004.

[23] J. Kurths and C. Zhou. Noise-enhanced phase synchronization of weakly coupled chaotic oscillators. Physics and Control, Jan 2003.

[24] T. P. Leung and H.-S. Qin. Advanced Topics in Nonlinear Control Systems. 2001.

[25] C. Li and X. Liao. Lag synchronization of rossler system and chua circuit via a scalar signal. Physics Letters A, Jan 2004.

[26] M. Lungarella, K. Ishiguro, N. Otsu, and Y. Kuniyoshi. Methods for quantifying the causal structure of bivariate time series. IJBC, 17:1–19, 2007.

[27] D. Orrell and L. Smith. Visualising bifurcations in high dimensional systems: The spectral bifurcation diagram. Int. J. Bifurcation and Chaos, Jan 2003.

[28] E. Ott. Chaos in Dynamical Systems. 1993.

[29] E. Ott, C. Grebogi, and J. Yorke. Controlling chaos. Physical Review Letters, Jan 1990.

[30] K. Park, Y. Lai, S. Krishnamoorthy, and A. Kandangath. Effect of common noise on phase synchronization in coupled chaotic oscillators. Chaos: An Interdisciplinary Journal of Nonlinear Science, Jan 2007.

[31] L. Pecora and T. Carroll. Synchronization in chaotic systems. Phys. Rev. Lett., 64(8):821–824, Feb 1990.

[32] L. Pecora, T. Carroll, G. Johnson, and D. Mar. Fundamentals of synchronization in chaotic systems, concepts, and applications. Chaos: An Interdisciplinary Journal of Nonlinear Science, Jan 1997.

[33] A. Pikovsky, M. Rosenblum, and J. Kurths. Phase synchronization in regular and chaotic systems. International Journal of Bifurcation and Chaos, Jan 2000.

[34] A. Pikovsky, M. Rosenblum, and J. Kurths. Synchronization a universal concept in nonlinear sciences. 2001.

[35] C. Pinto and M. Golubitsky. Central pattern generators for bipedal locomotion. Journal of Mathematical Biology, Jan 2006.

[36] A. Pisarchik, R. Jaimes-Reátegui, and J. García-López. Synchronization of coupled bistable chaotic systems: experimental study. Philos Transact A Math Phys Eng Sci, Aug 2007.

[37] A. Pitti, M. Lungarella, and Y. Kuniyoshi. Quantification of emergent behaviors induced by feedback resonance of chaos. Recent Advances in Artificial Life: Advances in Natural ..., Jan 2005.

[38] A. Pitti, M. Lungarella, and Y. Kuniyoshi. Exploration of natural dynamics through resonance and chaos. pages 558–565, 2006.

[39] K. Pyragas. Continuous control of chaos by self-controlling feedback. Physics Letters A, Jan 1992.

[40] L. Righetti, J. Buchli, and A. J. Ijspeert. Adaptive frequency oscillators applied to dynamic walking i. programmable central pattern generators. Proceedings of Dynamic Walking 2006 (this volume), Jan 2006.

[41] M. Rosenblum, A. Pikovsky, and J. Kurths. Phase synchronization of chaotic oscillators. Physical Review Letters, Jan 1996.

[42] M. Rosenblum, A. Pikovsky, and J. Kurths. From phase to lag synchronization in coupled chaotic oscillators. Physical Review Letters, Jan 1997.

[43] L. Smith. Chaos : a very short introduction. 2007.

[44] R. V. Solé, J. G. Gamarra, M. Ginovart, and D. López. Controlling chaos in ecology: from deterministic to individual-based models. Bull Math Biol, 61(6):1187–207, Nov 1999.

[45] J. C. Sprott. Chaos and time-series analysis. page 400, Jan 2003.

[46] S. H. Strogatz. Nonlinear Dynamics and Chaos. 2000.

[47] A. A. Zaher. Design of model-based controllers for a class of nonlinear chaotic systems using a single output feedback and state observers. Physical review E, Statistical, nonlinear, and soft matter physics, 75(5 Pt 2):056203, May 2007.

[48] C. Zhou, J. Kurths, E. Allaria, S. Boccaletti, R. Meucci, and F. T. Arecchi. Constructive effects of noise in homoclinic chaotic systems. Physical review E, Statistical, nonlinear, and soft matter physics, 67(6 Pt 2):066220, Jun 2003.