ETH Zürich/University of Zürich

Master's Thesis

# Emergent gait periodicity in evolved creatures on unknown terrains

*Author:*
Benjamin Ellenberger
09–919–622

*Supervisor:*
Prof. Dr. habil. Ruedi STOOP

*A thesis submitted in fulfilment of the requirements*
*for the degree of Master of Science in the*

Stoop Group/Institute of Neuroinformatics
Dept. of Information Technology and Electrical Engineering

Feb 29, 2015

# Declaration of Authorship

I, Benjamin ELLENBERGER, declare that this thesis titled, 'Emergent gait periodicity in evolved creatures on unknown terrains' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

ETH ZÜRICH/ UNIVERSITY OF ZÜRICH

# *Abstract*

Institute of Neuroinformatics
Dept. of Information Technology and Electrical Engineering

Master of Science

**Emergent gait periodicity in evolved creatures on unknown terrains**

by Benjamin Ellenberger

**Motivation.** The evolution of locomotion was of central importance for the spread of the first complex life in the ocean and the transition from the ocean to land. Locomotion is omnipresent in simple and complex organisms of micro and macroscopic level, thus there must exist an underlying natural process that simplifies its emergence. When observing a variety of locomotion gait patterns in nature, it becomes obvious how robust animal gaits are against perturbations. Therefore the underlying system has to be adaptive to the quickly changing environment locomotion takes place in. Unfortunately, our understanding of the fundamental process of adaptive gaits is still limited. In robotic applications, locomotion implemented using computationally expensive planning methods often fails to copy the delicate behavior of animals, especially when it comes to keeping locomotion robust and adaptive in unpredictable situations. The difference might be that robot locomotion often relies on a large sensory interface to sense the environment and an internal model to plan the next actuation of the limbs based on the current state of movement. Natural walking however relies on central pattern generators (CPG), which emit oscillatory signals to control and coordinate limb movement. It has been shown that the pyloric central pattern generators of the spiny lobster are closed neural systems that undergo chaotic oscillations [1]. We hypothesize that the underlying natural process to simplify the emergence of adaptive, oscillatory movement in locomotion patterns are chaotic systems, which are controlled into different periodicities and adapt their oscillation frequency based on feedback from the environment. The adaptive system consists not only of the central pattern generator, though; the whole complex of pattern generators, the limbs and the connecting joints all work together to self-adapt to the external constraints.

**Methods.** This work focusses on finding examples of locomoting, simulated creatures that use chaotic systems to control periodic leg movement. Therefore, an example chaotic system was chosen, the Chua circuit, to be controlled by a chaos control method called simple limiter control [2]. The limiting component, the simple limiter, can be set appropriately so that

the chaotic system presents chaotic and periodic trajectories. Furthermore, a simulator was implemented to evolve simple creatures in a rigid-body physics engine using an evolutionary process, a terrestrial flat plane environment, and a genome and phenotype structure to encode a morphology out of 3D primitives (such as boxes and capsules) connected by joints and a controller structure using the chaotic system. To compare the results of evolution using chaotic controllers, reference simulations were performed using a sinusoidal controller. Some preliminary experiments were performed to replicate the simple limiter control method and explore the influence of soft limiters and different limiter configurations on the behavior of the Chua circuit. Then a chaotic controller was configured with different limiters by the movement of the joint of a simple model leg. The model leg was set up in different settings: gravity-less space and interacting with ground, additionally with different joint torques and damping settings. Finally, the evolution simulator was run to evolve locomoting creatures on land using the chaotic controller.

**Results.** The simple limiter control method benefits from soft limiters in the sense that the simple limiter parameter ranges of different stabilized limit cycles are increased. Furthermore, under which conditions the simple limiter is applied to which part of the system plays an important role in the success of the control method and must be chosen wisely. In the simulation of the model leg, only limit cycles of periodicity 1 could be found. In the absence of gravitation, the model leg always converged to the limit cycle. In the interaction with ground, smaller limit cycles were stabilized, but the system tried to converge back to the uninfluenced larger limit cycle. It was found that damping leads to sub-manifolds of the original manifold described by the limit cycle. The increased joint torque leads to a larger limit cycle, however, contrary to expectations, does not make the system switch to a higher periodicity in either the controller or joint phase space. The evolution simulator successfully evolved creatures using both the sinusoidal and the feedback limited, chaotic controller. In the case of the creatures using the chaotic controllers, a more in-depth analysis of the trajectories showed that higher periodicity can be observed both in the controller and joint phase space. Furthermore it could be shown that the gaits generated by a chaotic controller are of high robustness towards perturbations.

# *Acknowledgements*

I would like to thank Professor Ruedi Stoop for his patience, numerous advices and kind critique. I am very grateful for all the support, discussions and material I was provided by the Stoop group, especially I want to thank Thomas Lorimer and Karlis Kanders for their help with the implementation of the chaotic model and limiter control, the interpretation of the data and the helpful discussions leading to different improvements of this thesis. In addition I want to thank my girlfriend Perrine Dubuis for her support during my studies and work on my thesis, without which I would not have had the same patience for the countless failed experiments and evolutionary runs without usable results. Furthermore, I want to thank Randolph Busch and Karlis Kanders for proof-reading this thesis and for finding inconsistencies in reasoning and explanations.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Locomotion to Conquer the World

Motion is a central aspect of life for organisms on earth. When the first life evolved in oceans and other open water, most organism motion was due to environmental forces such as wind or water streams. However, with the development of more complex, multicellular organisms, self-induced motion rapidly gained importance. Different species evolved a variety of ways of locomotion such as swimming or crawling, which permitted the spread of all faunal species across oceans. With the evolution of lungs and more resistive skin, the first animals escaped water and populated the coastal regions, which surely was one of the key events of life on earth. This was however impossible with the same gaits as in water, therefore the way animals locomote had to be adapted again. Today, anthropods (including insects and spiders), recognizable by their segmented body, make up $80\,\%$ of all described species and have conquered every environment on earth. With the exception of some primary producers that have found a way to spread completely without self-generated motion by exploiting environmental forces to distribute seeds, organisms of higher complexity such as vertebrates depend on locomotion, usually the most complex motor behavior the organism performs [3]. Locomotion is performed in different environments (e.g. air, water, ground), each with advantages and drawbacks to keep it stable. In water, fluid streams can prevent an organism from moving into one direction stably. On land, an organism faces problems with different obstacles and appropriate foot placement or switching between different gait patterns. Regardless of these problems, it seems that even very simple organisms could master the task of stable locomotion. The fundamental process of how robust, adaptive locomotion can so easily arise in many biological organisms is unfortunately largely unknown. Studies of the spinal cord suggest that a key aspect of biological, rhythmic motion is related to central pattern generators (CPG) [4]. The central pattern generator exhibits periodic and

chaotic oscillations [1] and is influenced through the aid of neuromodulation [5]. They serve different rhythmic oscillations, like a function generator, for many functions in vertebrate animals, such as locomotion, breathing, peristaltic and gastric movements or other oscillatory functions. In early studies by Brown et al., it was recognized that a CPG can control simple locomotion tasks even without the need of descending motor commands from the cortex [6]. Additionally, in the model of the Lamprey spinal cord, stimulation of the neurons can produce fictive swimming motion, indicative of a CPG neural circuit. It was proposed that there even was a basal CPG responsible for different kinds of swimming, crawling on a surface and burrowing in the mud [7], which could be altered using neuromodulators [8]. A similar CPG based mechanism was evolved to control a robotic lamprey [9]. An extensive review of robotic CPG applications can be found by the same author [10].

## 1.2 A Natural Approach

Nature seems to have solved the locomotion problem of getting from point A to point B extremely easily and elegantly. Even under constraints of rough terrain and obstacles interfering with the path, a solution is nearly always found and generally does not require long path and limb trajectory planning times. It seems as if the locomotion movements arose naturally from the interaction between the legs and the ground, because we rarely see an animal look at its feet while it runs. For such gaits to be stable, the locomotion must be very robust to disturbances. Yet, engineering solutions usually rely on longer planning times and still do not reach competitive solutions in terms of robustness and adaptability to the environment. Conventionally, the need for gait adaption has to be detected, before then computing a new limb trajectory according to the chosen gait. A new approach is to design a self-adaptive system that produces a variety of periodic movements and adapts the output trajectory through the interaction with the ground. Such a system would be computationally more efficient and the produced gait more robust than conventional methods. The interaction of the body and the controller with the environment automatically constrains the motion space of the legs in order to move the leg appropriately and make the creature move forward. The sensors from the joints as well as the weight, shape, and inertia of the body parts in this system act as limiters to the feasible gait motions.

### 1.2.1 Evolutionary Algorithms

Evolutionary algorithms mimic the general way of how evolution as a self-optimizing process finds competitive individuals able to survive on earth. The general approach is based on a population of individuals, each individual based on a genotype, an element defining the characteristics of the individual. The genotype is the blueprint of the animal and is

subject to variation through mutation and crossover. The fitness functions, which model the environmental constraints that define what has to be achieved in order to be called fit, are applied to all individuals of the population to measure their fitness. If an individual is competitive with respect to the fitness functions, it has a higher chance to reproduce than a non-competitive individual. The genotypes of individuals with high fitness values are then crossed, which forms a new individual with a genotype that is a mixture of its parents' genotypes. All individuals are subject to one or multiple types of mutations, which lead to slight changes in the genotype, thereby also leading to changes in the fitness of that individual. In order to keep the size of the population constant, some of the worst performing individuals are culled. Using this process, approximately optimal solutions to NP-complete problems can be found, such as the travelling salesman problem. This is due to the features of the variation operators, crossover leading to mixtures of different solutions which are possibly better solutions than the crossed solutions, and mutation helping to produce variation and to avoid getting stuck in local minima. The evolutionary algorithm therefore follows the scheme of a general gradient descent algorithm. It must be noted that the process per se does not aim to generate better individuals and that variational operations do not necessarily lead to a better fitness of an individual. However, the population under the evolutionary process will adapt to the fitness landscape, meaning that individual solutions will converge to global minima. Very fit individuals will arise from it, not necessarily showing the same or even similar solutions, depending on the complexity of the fitness landscape.

The locomotion task could be a challenging optimization task for evolution due to the very non-continuous nature the fitness landscape. A lack of symmetry for instance, constituting a simple change in some genome structures, can instantly make the locomotion performance drop. For example, a loss of one of two symmetric limbs would make it much harder to walk properly due to imbalance. Therefore, the gradient on the fitness landscape might be of little help in some areas, degrading the algorithm to random sampling. Additionally, the representation of a locomotion solution might require a large parameter space, which slows down the search for good solutions. Despite these restrictions, experiments show that the evolutionary algorithm is still applicable and finds valid solutions (see sections 4.1 and 4.2.2.2).

### 1.2.2 Chaotic Systems as a Source of High Variability

Chaotic systems (introduced more thoroughly in section 3.3.1) are the underlying model of many different physical processes such as the three-body problem in astronomy studied by Poincaré [11], population models in biology studied by Verhulst [12], weather models in meteorology studied by Lorenz [13] and many other models from different fields. Before the beginning of chaos theory dated back to the 19th century, it was thought that approximate

knowledge about the initial state of a deterministic system was sufficient to approximate knowledge about the future. In natural sciences, the exact knowledge of a real-world state of a system is impossible, as the real-valued nature of a natural, continuous system state has infinite precision. However, an experiment can only reveal a limited-precision state of the system. Chaos theory showed that a small variation of the initial conditions can cause a large variation in the future state, as already Poincaré put it when studying the three-body problem. The former belief that all deterministic systems can be perfectly predicted using approximate knowledge turned out false for most systems [14]. Instead it was found that such unpredictable systems had to be studied much more in-depth to understand; when different initial conditions converge to the same solutions, and what conditions lead to a variety of different outcomes. It is the property leading to the unpredictability, namely the sensitivity to the initial conditions, that can be used to produce a high variability of trajectories. In a chaotic system, the trajectories are guided by an infinite amount of unstable periodic orbits (UPOs), that, in interaction with each other, lead to the variability of trajectories. Using a chaotic system and a chaos control method together with an evolutionary algorithm, the evolutionary algorithm could tune the system's output to show a desired solution trajectory to simplify the emergence of robust locomotion. As it turns out, simple limiters, introduced in the next section, are a very natural way of influencing chaotic systems. Through their innate coupling with the chaotic system, because they express limitations in the state space in the same 'language' as the system and are hence elegant by simplicity, simple limiters could be the key to control the system into different trajectories for periodic and chaotic movement.

### 1.2.3 Simple Limiter Control

Simple limiter control, first introduced in the paper by Ned J. Corron et al. in 2000 [2], is different from other preceding chaos control strategies. The OGY method described by Ott, Grebogi and Yorke [15], one of the first chaos control methods at hand, states that small, wisely chosen perturbations applied to the system can stabilize a particular UPO. The perturbation must be applied once per cycle, but the chaotic system must be intensively studied through an appropriate Poincaré section to understand what the precise perturbation must be. The time to measure, generate and apply a control signal, called the latency of the OGY method, is another limitation that allows only for a certain range of frequency control. The simple limiter control method is much more simple. A limiter expresses a limitation in the state space of the chaotic system and thereby suppresses certain periodic orbits. Interestingly, however, the remaining reachable periodic orbits are being stabilized. The seemingly unpredictable trajectory always deterministically switches at the same point to the same periodic orbits in the phase space. Using the simple limiter, the switching can

be prevented at some specified point and leave the system state no other choice than to stay on its current orbit and to repeat its trajectory, and hence the state is stabilized on a trajectory of a certain periodicity. In 2001, Wagner and Stoop showed that the optimal simple limiter control ranges for different limit cycles can be found using the derivative of the system [16]. In a natural system such as the actuated leg joint, this limiter can be much more subliminal, be it the weight or dimensions of the body parts or the limits, damping or friction of the joints, or even the fact that two physical objects can not interpenetrate each other.

### 1.2.4 Experiment Outline

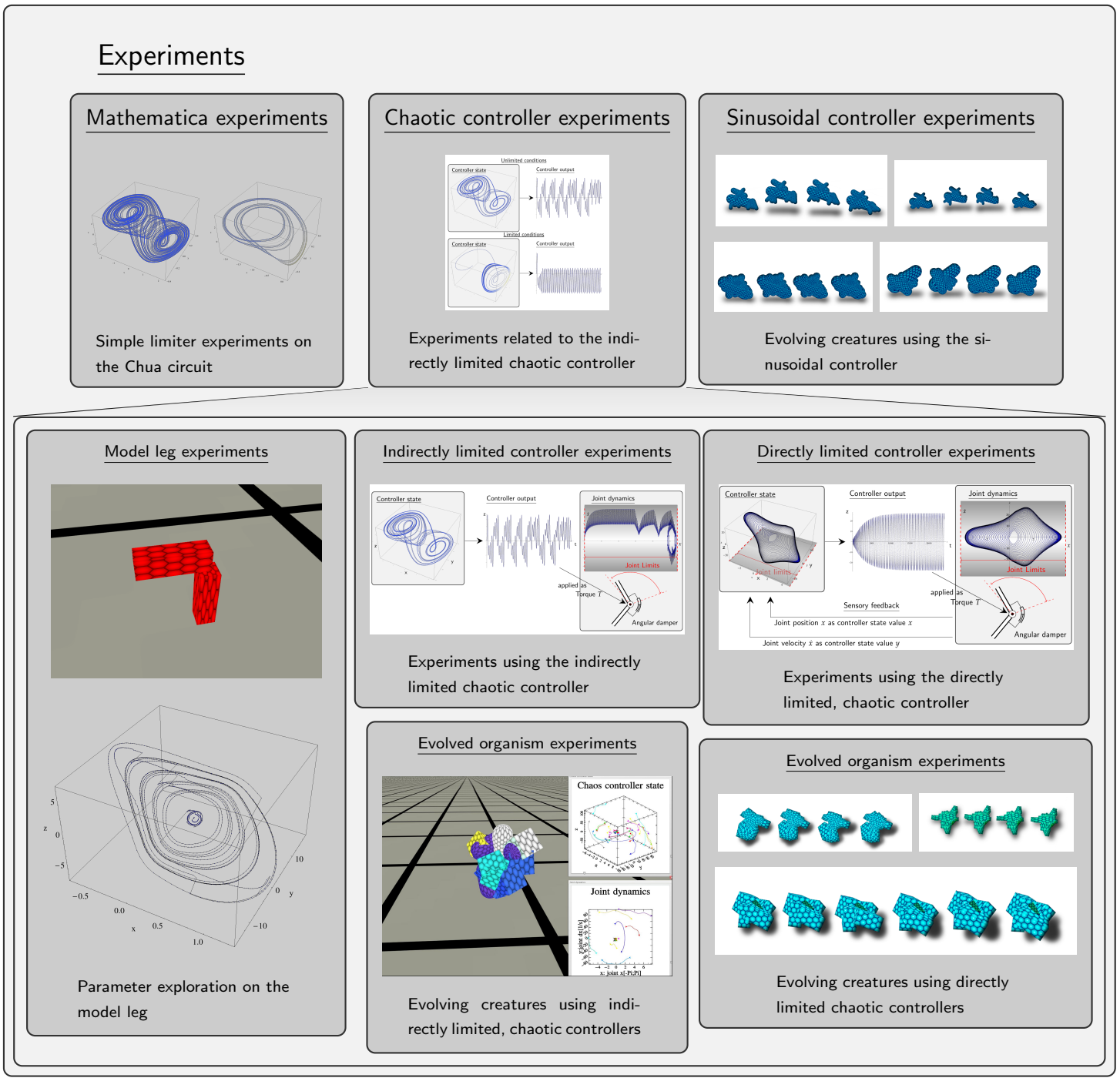


FIGURE 1.1: The chart of experiments performed for this thesis. First, the simple limiter control method was used on a model chaotic system called the Chua circuit to understand the influence of the limiter and the limiter softness. Then, the first experiment on the evolution simulator was run to generate locomoting creatures using sinusoidal controllers for the gait pattern. Subsequently, two chaotic controllers were developed, one without feedback (indirectly limited controller) and one with sensory feedback of the controlled joint (directly limited controller). Both were first tested on a model organism called the model leg, then were used to evolve locomoting creatures. While the indirectly limited controller did not produce any periodic gait patterns due to the inability to limit the internal chaotic system, the directly limited controller could produce periodic gait patterns, hence, locomoting creatures could be evolved.

The goal of this thesis is to use evolutionary algorithms and the above mentioned, chaotic control system to evolve creatures in a rigid-body physics engine to find example creatures showing basic locomotion abilities emerge in the interaction with the ground. Therefore, a simulator is developed to evolve artificial creatures using a specially tailored type of genome to co-evolve the individual's controller and morphology (see Chapter 2). In a first evolution, the creature's movement is controlled by an engineered controller emitting sinusoidal leg movement signals, so that each joint degree of freedom moves according to the sinusoid's parameters of amplitude, frequency and offset (see Chapter 3 and 4). After successful evolution of locomoting creatures, an example chaotic system is chosen and a simple limiter is applied to it to observe the influence of different simple limiter configurations (see Chapter 3). Subsequently, creatures are evolved being driven by chaotic leg movement controllers using the previously chaotic system to exert joint torques (see Chapter 4). In case the controller can not be integrated into a locomoting creature because the evolutionary algorithm is unable to limit the controller using the morphology only, a controller using sensory feedback to represent the limiter in the controller space will be developed (see Chapter 4). Using this direct, feedback limited controller, the morphological limits of the creatures can directly influence the controller behavior to adjust its stabilized limit cycle. If a morphology/controllers configuration can be found that imposes the right limits onto the controllers so that they stabilize limit cycles which emerge into a locomotion gait, the stability and adaptivity of the gait could be superior to general control methods. The superiority could be explained through the inherent self-stabilizing nature of the limit cycles and the self-adapting nature of the chaotic system in the presence of simple limiters. Thus, the control system is robust to small disturbances and gait-adaptive in case of insurmountable obstacles.

# Chapter 2

# Evolutionary Optimization

The following chapter describes the evolutionary optimization algorithm as it is implemented in the Minemonics simulator. The Minemonics simulator is the experimentation platform used throughout this thesis and is a project that was developed from scratch, mainly because other simulators for evolving virtual creatures are unpublished by the respective research groups, are outdated, or implement a different type of simulation than is needed to simulate the model described in this thesis. Apart from the research interests, the simulator was made to be extensible for other research experiments to be implemented. It is available open-source from its git repository on Github [17].

## 2.1 Basic Components

To understand the purpose of the simulation, it is important to first understand its different components. Therefore, the basic components of the simulator and their respective features are discussed first. Components are described from coarse to fine grain, starting with the universe and ending with the genomic and phenomic structure of a simulated individual.

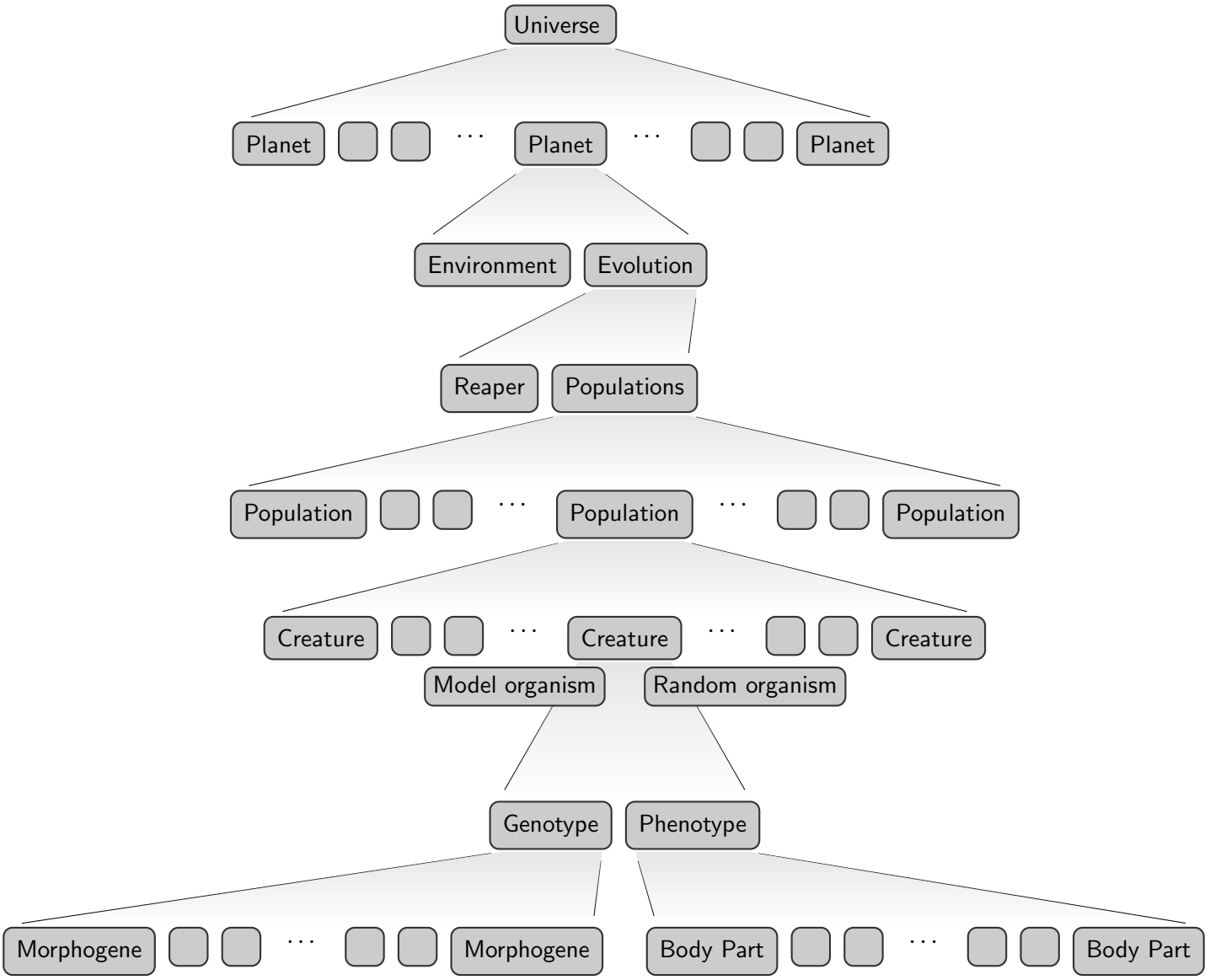


FIGURE 2.1: A figure of the universe and its components.

### 2.1.1 Universe, Planets and Environments

The top-most component of the simulation is its own universe. The simulator's universe is a set of different planets, each planet being a certain setting of evolutionary run. A certain planet consists of an environment, an instance of evolutionary process and a number of epochs. The environment is a representation of a plane environment, but could also be a hill terrain or a deep-sea water environment. For the experiments in this thesis, originally an additional hill environment was planned, but was then abandoned and only the plane environment was used. The type of evolutionary process defines, how many creatures of the planet are evaluated simultaneously in the same environment, and the percentage of

creatures that are culled, varied and sown. Within the bounds of this thesis, only single creature evaluation was performed.

#### 2.1.1.1 Planetary Physics

Each environment defines the type of planetary physics simulation. The one type of environment, flat plane, simulates the terrain and the creatures using newtonian mechanical physics. To do that, the simulator uses the Bullet Physics engine[18] to simulate a terrain for the creatures to move on and builds the creatures from different rigid body primitive types (such as boxes or capsules) coupled by joint constraints with different configurations. The Bullet Physics engine is a numerical classical mechanics simulation engine, featuring rigid body as well as soft body physics. Bodies can be constrained using various types of constraints with constraint limits, springs, dampers and motors. The constraints are numerically solved using one of the many types of available solvers with different convergence speeds and accuracies of result. The most robust results were achieved using the Featherstone Multibody solver, the reason for the robustness being discussed in section 2.1.4.4. To date this model only supports single degree of freedom joints, however this does not affect the evolutionary optimization much. Three degrees of freedom joint can easily be approximated by three one degree of freedom joints, as long as the 3 1-DoF joints do not gimbal-lock i.e. two of the degrees of freedom fall together on one.

### 2.1.2 Epochs

An epoch of a certain planet models the changes of environmental factors, which induce a change of the fitness landscape on the planet. What is considered fit in one epoch might not be fit in another, therefore creatures have to adapt when the fitness changes. In the simulation, an epoch is represented by a set of juries (fitness functions) and a transition condition, which, when met, ends the epoch and starts the next epoch. In the simulator, only one single infinite epoch was used, since there was no need in the experiments to change fitness conditions.

#### 2.1.2.1 Juries

Juries or fitness functions are part of an epoch in which they are active. A jury collects data on a creature performance during its evaluation and finally rates the performance according to its definition of fitness. Commonly used juries of the simulator are the individual's average escape velocity with respect to the origin (*AverageVelocityJury*), the average height above

ground of all limbs (*AverageHeightJury*) or the number of ground contacts during the evaluation (*ContactsQtyJury*). Additionally, each jury has a combination weight, which indicates the weight of this performance rating in the total performance evaluation of the creature, and a sorting direction, which indicates whether a higher rating is considered better or worse than a lower rating for this type of fitness value. Fitness values of different types of juries then are combined as a weighted sum of linear ranking scores, where the scores are gained from competitive performance rankings of all creatures of one population, once for each jury type. The culling procedure will be described in section 2.1.6.1. The following paragraphs describe the four common jury types of the simulator.

**Average Velocity Jury** The average velocity jury measures the displacement of the creature from the origin after the performance, divided by the evaluation time. Additionally, the measure is scaled by the edge length of the total volume cube (Equation (2.1.4)) of the creature to achieve a scale-invariant measure as described in the section 2.1.4.

**Average Height Jury** The average height jury measures the average height for each creature's limbs when performing. This jury inhibits creatures that grow very tall and then only fall over or creatures exploiting unrealistic torques in order to get a high fitness can be inhibited. This measure is scaled by the edge length of the total volume cube (Equation (2.1.4)) of the creature to achieve a scale-invariant measure as described in the section 2.1.4.

**Ground Contact Quantity Jury** The ground contact quantity jury measures the number of limbs that come in contact with ground during the performance. This jury is meant to reduce the number of contact points to enforce the usage of legs and inhibit rolling creatures.

**Ground Contact Ratio Jury** The ground contact ratio jury measures the number of limbs that come in contact with ground during the performance divided by the total of limbs of the creature. This jury, in contrast to the ground contact quantity jury, does not limit the number of ground contact points in absolute terms, but allows more complex creatures to have more contact points.

### 2.1.3 Populations

A population of creatures in the simulator lives on a certain planet. Depending on the evolution settings of the planet, one creature, multiple creatures of one population, multiple creatures of multiple populations or multiple whole populations are evaluated at the same time within one evaluation slot. A population is initialized with a certain initial number of

individuals. The individuals can be randomly generated creatures or model organisms, the latter being used for experiments with known morphology and controller settings. During the evaluation of a population, the number of individuals generally falls below the original population size, because several individuals are culled early during the evaluation. This happens when creatures surpass boundaries of physical realism (a main source of this is when creatures generate high joint torques that lead to very fast limb velocity, leading to unrealistic torques that lift creatures off the ground and make them fly). Therefore, the simulation culls them early to remove them from further consideration early on.

### 2.1.4 Creatures

A creature or individual is part of a population and is subject to evaluation. The creature is based on two main components, its genotype and its phenotype. Similar to biology, the genotype is the compact blueprint of the creature. Developed from it is the phenotype, which is the explicit form of the creature, representing the body of the creature in the physics engine and the controller in its fully wired form. Furthermore, we define two important measures for creatures, which are its total body volume and its size. The total body volume is just the total volume of all limb primitives. The size of a creature $C$ with number of limb primitives $N(C)$ is measured as being the third root of the total volume, denoting the edge length of the total volume cube.

$$size(C) = \sqrt[3]{\sum_{i=1}^{N(C)} V(p_i)} \tag{2.1}$$

The size measure is most important, because it is used to scale the fitness functions involving distances. Thereby it is possible to make a distance travelled independent of the creature size, meaning that a larger creature gets the same fitness for travelling a larger distance as a smaller creature for a smaller distance, if the ratios between creature size and respective travelled distance are the same for both creatures.

#### 2.1.4.1 Genome

In the traditional approach on genomes, a fixed length string of bits is used to encode higher order data types such as integers or doubles or data structures such as trees or matrices. However, many state-of-the-art genomes deviate from this original representation by directly using higher order representations and thereby avoid the usually hard to solve representation problem ('What representation should be chosen so that every change in the bit string

leads to a new, valid representation?'). This more modern approach was also chosen in the Minemonics simulator, using an adapted version of the genotype described in [19].

The encoding structure of the genome is one of the base structures of the simulator. It encodes both morphology and the controller of the creature. Inspired by biological DNA, the genome of the creature uses an indirect encoding. As opposed to direct encoding, indirect encoding does not encode every element of the phenotype explicitly, but uses element classes and branches between them to encode their relations. The advantage of indirect encodings is that an indirect coding evolves element classes instead of element instances, which leads to a higher abstraction of phenotype building blocks within the genome. A sequence of element classes, forming an abstract, higher order feature such as a type of leg, can thus be repeatedly expressed within a creature morphology in different scalings and symmetries. The classes therefore help to evolve similar, repeated, morphological elements such as limbs of different sizes or symmetrical leg pairs, which all use the same leg definition in the genotype. To visualize the difference between an indirect and a direct encoding, an example creature is represented once in indirect encoding (Fig. 2.3) and once in direct encoding (Fig. 2.4). In the phenotype section, the creature's morphology resulting from both encodings is shown (Fig. 2.5).

The genome defines a directed, possibly cyclic graph, whose nodes are called morphogenes, defining the classes of limbs, and whose edges are called morphogene branches, defining the joint relations in between the limbs. A morphogene definition (Fig. 2.2) consists of the primitive type of the limb (a cuboid or capsule), the limb's dimensions, its base orientation in space, material properties such as friction, restitution and color, and a direction for the joint anchor. The joint anchor is a three dimensional vector and represents a position on the surface of the primitive type which denotes, where the limb defined by the morphogene class will be attached to the parent limb through a joint in the final phenotype. Furthermore, the morphogene contains a number of morphogene branches and influence factors related to how the branches are influenced by the limb. The influence factors in the morphogene are the repetitions limit, which limits the number of repetitions of that morphogene along one top-bottom branch, and the segment-shrink factor, which shrinks all dimensions of the morphogenes branched to from this morphogene.

A morphogene branch (Fig. 2.2) consists of the following joint properties.

The pitch and yaw axis define two of the three orthogonal joint rotation axes, the third being implicitly defined by the cross product.

The joint rotational limits for the pitch, yaw and roll degree of freedom limit the rotation to a certain range.

The joint damping coefficients for all degrees of freedom cause the joint to be damped, the coefficients representing $c$ in the damping force equation $F = -c \cdot \dot{x}$.

Furthermore, the morphogene branch contains a controller gene and some additional influence factors. The influence factors in the morphogene branch are the flip and the mirror flag, responsible for two types of symmetry, that are described in the context of the embryogenesis in section 2.1.5.

A controller gene is either a sinusoidal controller gene or a chaotic controller gene. The sinusoidal controller gene defines a controller emitting a sinusoidal signal based on x- and y-offset, a frequency and an amplitude. The chaotic controller gene defines a chaotic controller, through the type of chaotic system, the initial conditions of the system, and the integration step. The resulting sinusoidal and chaotic controller are described in their respective sections 3.2 and 3.3.

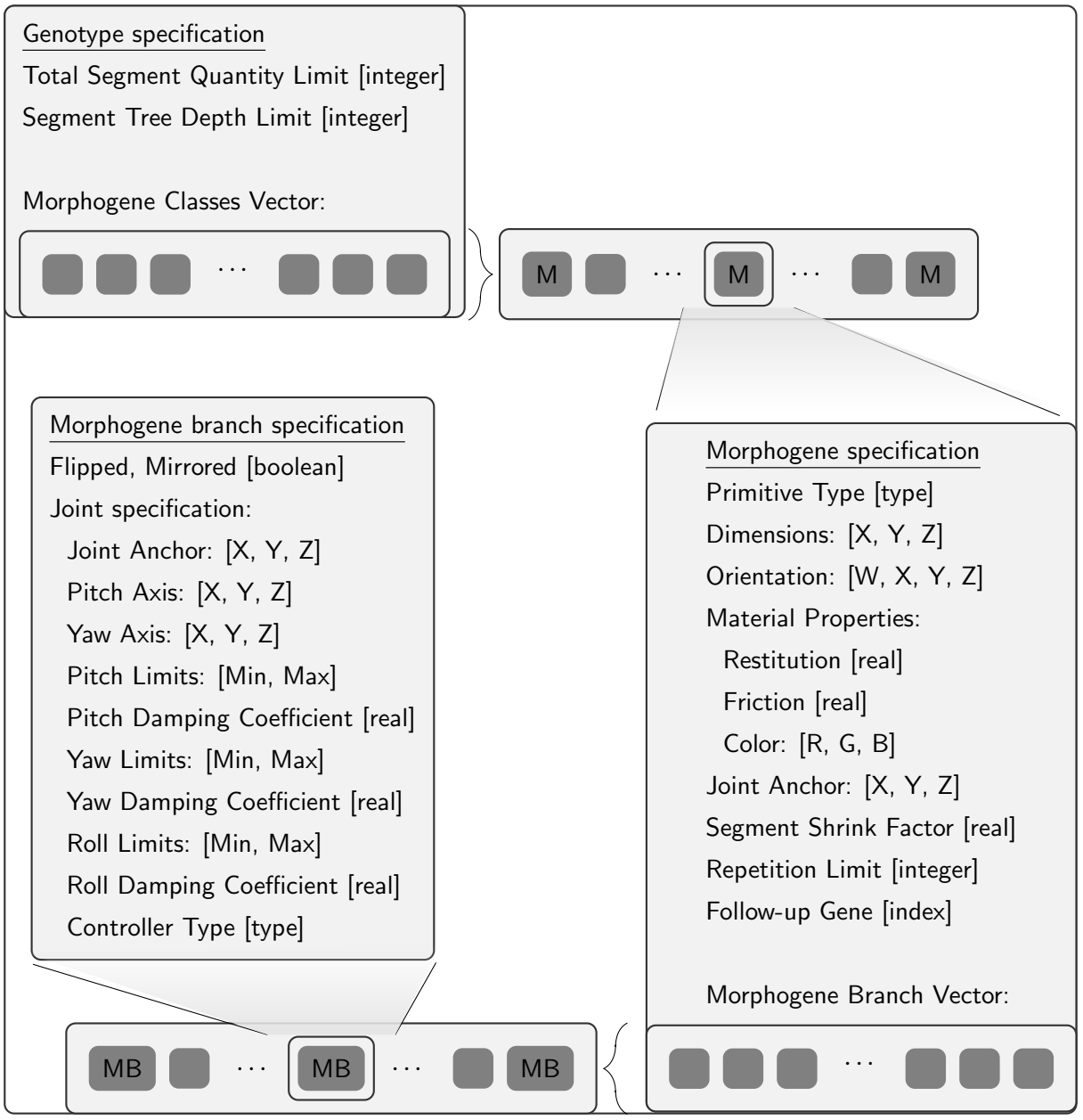


FIGURE 2.2: The structure of the genome, morphogene and morphogene branch. A morphogene defines a certain class of limb and a morphogene branch defines a certain class of joint to connect two limbs.

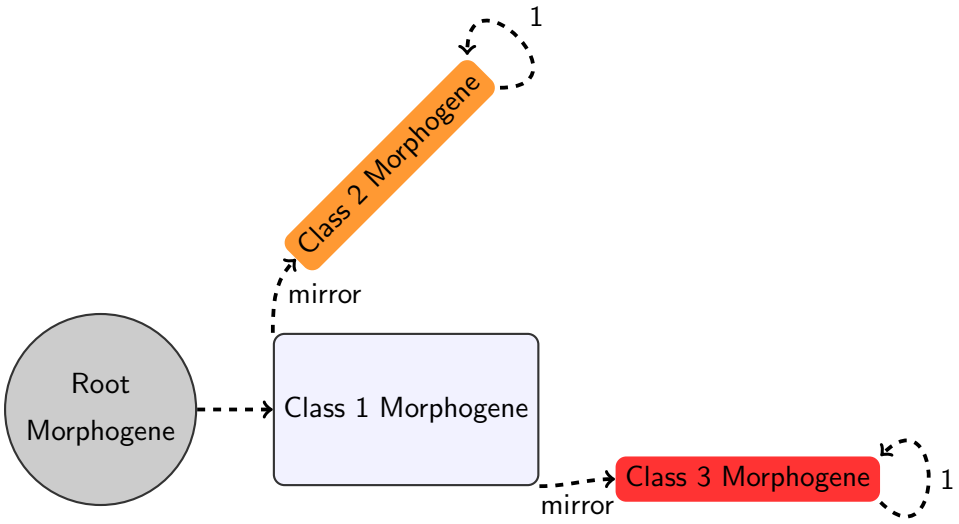


FIGURE 2.3: An example creature using the indirectly encoded genome. The number at the branch denotes the number of repeated child body parts of the type it points to which are to be attached to the respective parent body part of the phenotype. Mirror means that the following branch will be mirrored along the primary axis of the parent body part.

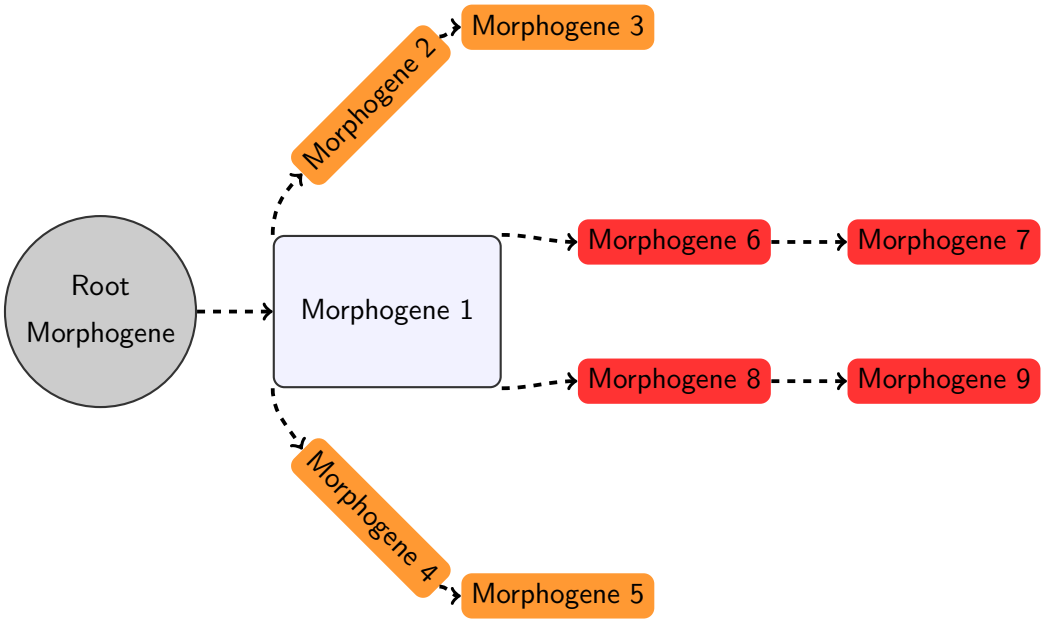


FIGURE 2.4: A figure of the same example creature using a directly encoded genome. The mirror flag and the repetitions do not exist in this type of encoding, therefore the genotype is basically an explicit representation of the phenotype.

#### 2.1.4.2 Phenome

Phenomes are the fully developed versions of genomes and are represented as a tree. In traditional genetic algorithms, genotypes usually do not have a development stage where

the genotype is transcribed into a phenotype because the traditional genome can be directly evaluated. In modern approaches however, it is often impossible to evaluate the individual directly based on the genotype, especially when the genome structure uses indirect encoding as is the case in this simulation. Therefore, the genotype first has to be developed through a process called the Embryogenesis.

In the Minemonics simulator, two different rigid body constraint models were evaluated for their stability when used within an evolutionary process. Both phenome models are based on the same representation, but the important difference is the underlying numerical physics simulation model. One of them, called Constrained Rigidbody Phenotype representation, reduces the degrees of freedom of the bodies with constraints, the other uses a Featherstone Multibody representation, increasing the degrees of freedom with mobilizers. The main issue to be resolved was that the parameter ranges in the Physics Engine were not completely clear, which means that some lead to invalid definitions of one or multiple joints, causing an instable simulation of the individual. Instability here means that a single invalid joint definition leads to convergence problems in the solver, so that the joint is constantly twitching. Secondly, instability can be caused by an individual strained by tensions, because the numerical solver is unable to solve multiple contradicting constraints, and therefore applies high forces and torques to resolve the constraints despite obvious contradictions. This issue could be fixed by using the Featherstone Multibody representation.

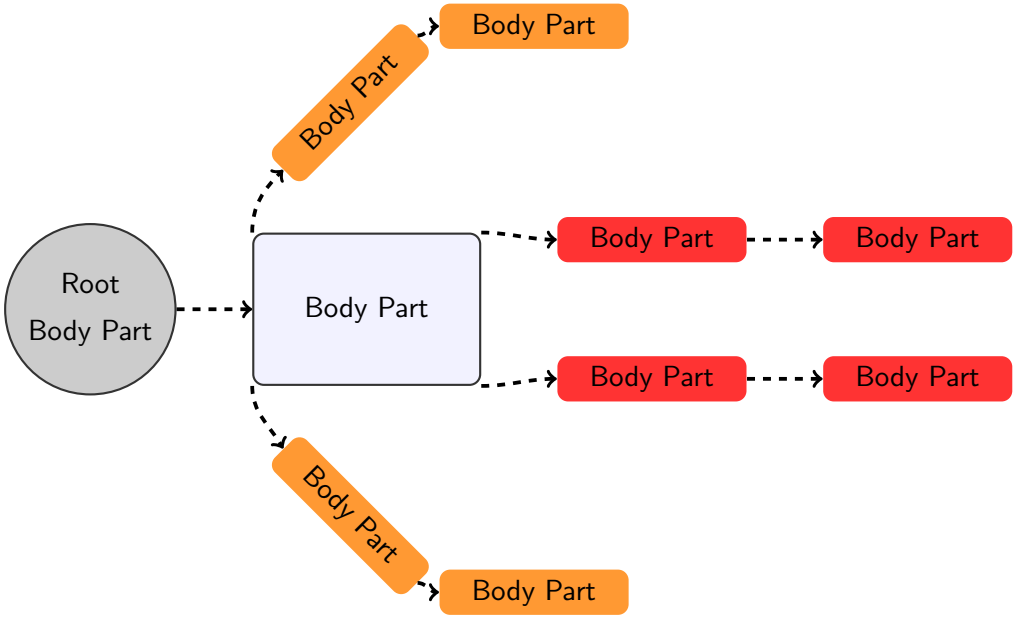


FIGURE 2.5: A figure of the phenotype of the example creature when transcribed by the Embryogenesis.

#### 2.1.4.3 Constrained Rigid-body Phenotype

The Constrained Rigid-body Phenotype model is built from rigidbody primitives and typed joint constraints. Every rigid-body has a full representation of its own position and rotation (Figure 2.6), which is then influenced by gravity, other rigid-bodies and the joint constraints. A joint constraint constrains two rigid-bodies by defining the center of rotation (CoR) of the joint in both body part coordinate reference frames, and then tries to keep both CoR definitions overlapping in the world reference frame under the additional constraints of joint limits and other influences. Many joint constraint types with different features exist, mainly because some constraints have a simpler constraint row representation than others. Generally, it is a good strategy to use the most lightweight constraints over the the computationally intensive constrains in order to get a fluid simulation. For the constrained rigid-body phenotype, many different constraint solvers exist, which either solve the constraint groups directly or in a numerical fashion using a number of iterations. The following solvers are implemented in the Bullet Physics engine [20]:

- Sequential Impulse Solver (iterative, very instable when linking body parts of large mass ratios with joints)
- Projected Gauss-Seidel Solver (iterative, bad convergence)
- Mixed Linear Complementarity Problem (MLCP) Solvers
  - Dantzig Solver (direct, high computational demand and singularity issues)
  - Lemke Solver (direct, does not always converge)
  - Non-linear Non-smooth Conjugate Gradient Solver (iterative, better than PGS, but only at high number of iterations)

The solvers all result in very stable simulations of complex, hand-made phenotypes. To understand why issues arise when using them with evolutionary algorithms, a quick overview of the internal numerical solver procedure has to be given.

When representing the morphology of the creature using maximal coordinates, a creature body is defined using rigid bodies and pairwise constraints. In that case, the constraints are resolved separately, returning the change in velocity for each body, then correcting impulses have to be applied to the involved rigid-bodies and finally the velocity and position update is computed for every rigid-body individually using the single inertias of the rigid bodies. With this type of solver, problems arise when evolution comes into play, because it evolves and

evaluates creatures from the whole large search space, resulting in very diverse individuals. Therefore, very invalid or strongly contradicting constraint groups are composed. Since in iterative constraint solvers, the constraint rows are not solved all at the same time but sequentially, the contradicting constraints then apply high forces and torques to the single rigid-bodies to resolve the invalid constraint or contradicting constraints, leading to pseudo-forces, that produce unrealistic physics movement. Therefore, the Constraint Rigidbody Phenotype was abandoned and the Featherstone Multibody Phenotype model was developed. It might be, that using a more constrained genotype representation would have lead to only valid and non-contradicting constraints, but such a representation could not be found during the time of this Thesis. A more in-depth discussion of the different solvers can be found in [20].

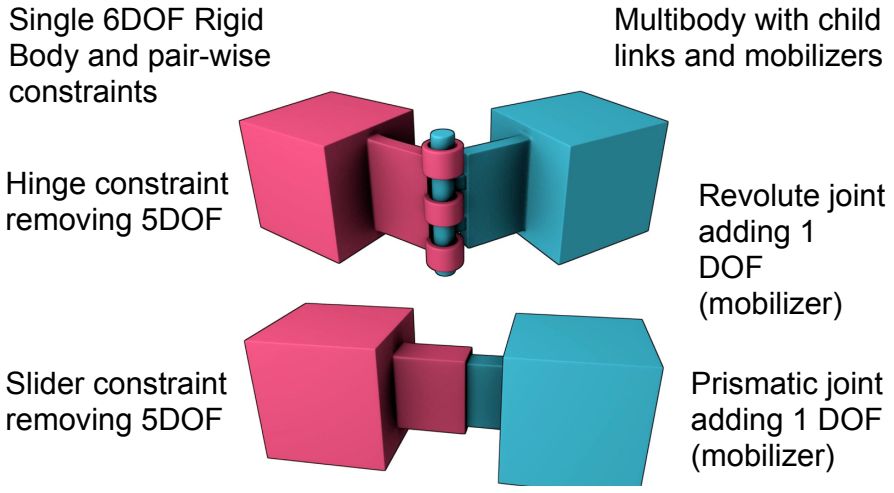


FIGURE 2.6: Maximal vs. reduced coordinates. The text describes how a complex structure, and a hinge and slider constraint are represented in maximal (left) and reduced (right) coordinates. The Constrained Rigidbody Phenotype uses maximal coordinates, the Featherstone Multibody Phenotype uses reduced coordinates. Image from [20].

#### 2.1.4.4 Featherstone Multibody Phenotype

The Featherstone Multibody Phenotype is based on a fundamentally different concept called reduced coordinates or generalized coordinates (Figure 2.6). Reduced coordinates express only the degrees of freedom of each rigid-body relative to its parent body part, starting from each joint not having any DoFs. A simple hinge joint is then expressed as a 1-DoF revolute joint angle. Instead of constraint, as in the Constrained Rigidbody Phenotype, we call such a multibody joint a mobilizer. In the case of reduced coordinates, the position, velocity and acceleration are expressed with reference to the parent body part, meaning that we start to apply position and velocity updates at the root element and continue with the child body part. We also calculate the inertia of the whole hierarchy instead of only the inertias of the interacting body part when resolving body part-pairs. In the Featherstone representation,

it is not necessary to resolve constraints, therefore we do not experience resolving forces and torques, thus no gaps exist between the rigid-bodies. The Featherstone representation, however, can not deal with circular structures [20], but such a feature was not needed in this simulation. This phenotype is very robust when it comes to hand-made as well as evolved phenotypes. That is why it is more suitable for evolving virtual creatures.

#### 2.1.4.5 Model Organisms

The model organisms are hand-coded genotypes to be developed into creatures that have specific properties. They are mainly used for testing and experimentation purposes, because they feature a non-redundant, simple genomic description and well-defined body part and joint descriptors.

**Model Leg** The model leg is a simple creature built from two equal body parts and one joint. The joint can be either set to a one DoF hinge joint or a three DoF spherical joint. The model leg's main purpose is to run experiments on chaotic controllers on a simple creature to observe the controller's behavior when changing gravity, restitution and friction.

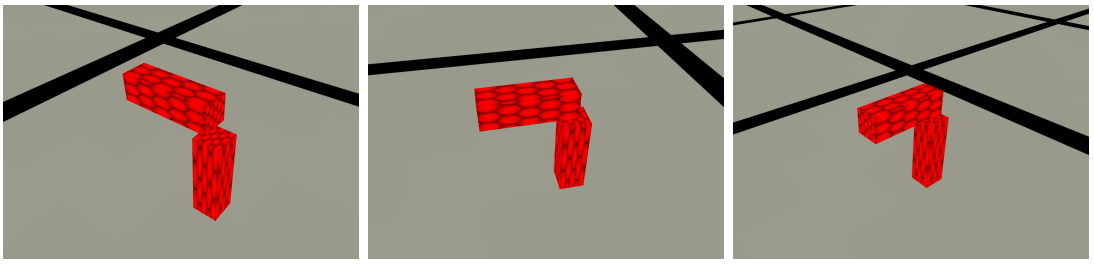

FIGURE 2.7: The model leg, one of the model organisms of the Minemonics simulator. It features one single joint connecting two body parts.

**Snake** The snake is a creature built from a chain of equal body parts connected with one or three DoF joints similar to the model leg. In fact, the snake is just a chain of model legs with a higher number of body part-joint repetitions.

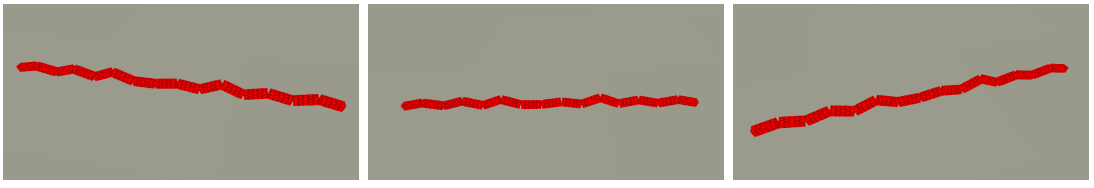

FIGURE 2.8: The snake. It features one long chain of body parts connected by either 1 DoF or 3 DoF joints.

**Pod** The Pod is a creature that can be configured to have a certain number of legs and a certain number of body elements. The main body is a body part on which the legs are attached in a circular manner. The number of legs is divided by the number of bodies, so that the same number of legs is attached to each body. With the same definition, its is possible to build insect-like creatures such as bugs, spiders, caterpillars and centipedes. It is mainly used to debug the Genotype-to-Phenotype transcription (Embryogenesis). Furthermore, experiments with a chain of chaotically oscillating controllers were run on it to see if patterns would emerge from it if the oscillators get coupled through the interaction with the ground.

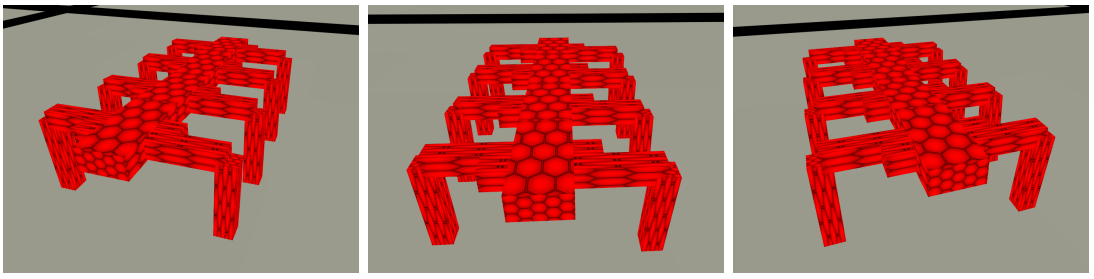

FIGURE 2.9: The pod is a basic structure that can be configured into having a certain number of main bodies and a certain number of legs. Thereby different morphologies such as hexapods or quadrupeds can be easily created.

**Ragdoll** The ragdoll blueprint produces a human-like form. It consists of differently configured joints and is mainly used to debug the Genotype-to-Phenotype transcription (Embryogenesis).

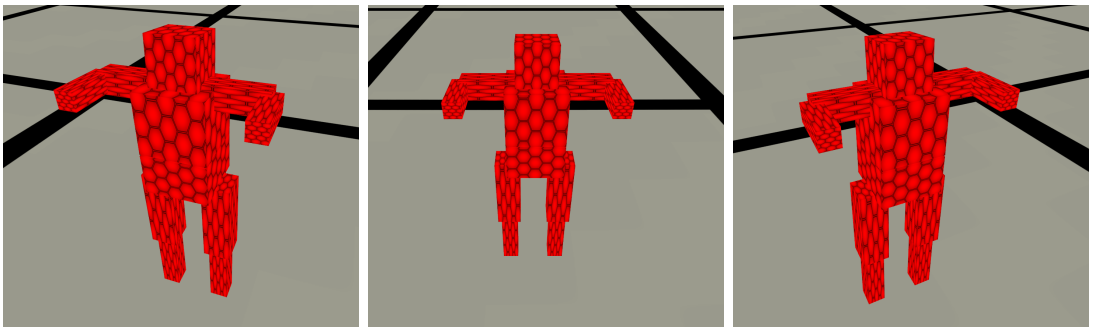

FIGURE 2.10: The ragdoll model organism. Its joints are differently configured so that errors during debug time of the Embryogenesis are revealed.

### 2.1.5 Embryogenesis

The Embryogenesis of a creature is a straight-forward process of transcribing a genotype into a phenotype. Starting from the root morphogene, the genome is expanded in a breath-first-search manner. This means that the process starts at the root node and transcribes

the root's child nodes first, before moving on to the higher degree children. To store the intermediate results of the generation process, the Embryogenesis uses phenotype generators. Intermediate results are the morphogene branch of the parent morphogene and the parent body part element, the position and orientation of the current generation process, the current shrinkage factor of the generated body parts, if this is a flipped or mirrored branch, the length of the current root to leaf path and a list of the number of repetitions of different morphogene classes along this branch. The Embryogenesis is influenced by so called influence factors, which are propagated along the top-bottom path of the phenotype-tree. The influence factors, as mentioned before, are the flip and mirror flag of the morphogene branch and the current shrink factor, the maximum repetition limits, and the total number of body parts of the morphogene. The flip flag causes the embryogenesis to transcribe a top-bottom genome branch twice in the phenotype, once in its original form and once in flipped form. Flipping means that all positional and orientational definitions of the branch and following body parts are negated to create a second flipped version of the original genome branch. The mirror flag causes the embryogenesis to transcribe the genome branch twice in the phenotype, once in its original form and once a mirrored form mirrored the primary axis of the body part the branch is attached to. The current segment shrink factor shrinks all body part dimensions of the morphogene the branch points to and propagates along further branches of that morphogene. The maximum repetition limits, one per morphogene, permit or deny a further repetition of a body part along this top-bottom path of the phenotype tree. The total number of body parts causes the Embryogenesis stop when reached by the current body part quantity.

The following pseudo code 2.11 describes the procedure of the embryogenesis.

**Inputs:** Genotype *genotype* of the individual
**Result:** Fully developed morphology and controller of the creature

```
get rootMorphogene from genotype
phenotypeGenerator.morphogene = rootMorphogene
phenotypeGenerator.segmentsDepthLimit = 0
currentBodyPartQty = 0
phenotypeGenerator.baseOrientation = Identity
phenotypeGenerator.repetitionsList = {}
add phenotypeGenerator to generatorsList

while generatorsList is not empty
  get generator from generatorsList

  exit if generator.segmentsDepthLimit = genotype.segmentsDepthLimit or
    currentBodyPartQty = genotype.maxBodyPartQty

  build new childBodyPart according to generator.morphogene, generator.flipped
    and generator.mirrored

  if childBodyPart has to be attached to a parentBodyPart
    append childBodyPart to parentBodyPart
    build joint & controller according to generator.morphogeneBranch

  rotate generator.baseOrientation with childBodyPart.orientation

  get morphogeneBranches from generator.morphogene

  for each branch in morphogeneBranches
    if branch.morphogene not exhausted in generator.repetitionsList
      create new phenotypeGenerator or multiple if branch is flipped or mirrored
      copy phenotypeGenerator settings from generator
      phenotypeGenerator.segmentsDepthLimit += 1
      phenotypeGenerator.repetitionsList[branch.morphogene] += 1
      phenotypeGenerator.morphogeneBranch = branch
      add phenotypeGenerator to generatorsList

  currentBodyPartQty +=1
```

FIGURE 2.11: Developing a creature using Embryogenesis

### 2.1.6 Reaper

At the end of all evaluations for a certain population's generation, the simulation arrives at the evolution step. In the evolution step, the following operations are performed by the reaper: Culling a percentage of the worst performing creatures according to the total fitness ranking, variation of the remaining creatures with different variance operators according to the respective percentages and sowing of new creatures partly from crossover and random generation. In the Minemonics simulator, 30 % are culled first. The remaining creatures

are equally distributed among the variation operations Gene specification, Gene split, Gene purge, Branch specification, Grow stub gene and graft feature mutation, another equal share from the fittest creatures is left untouched following an elitism strategy. Then the missing 30 % are replaced with equal amounts of creature generated by crossover and randomized instantiation.

In the following sections, the different operations are described in detail.

#### 2.1.6.1 Culling

In order to cull individuals from the population, the individuals first have to be sorted according to their fitness. For a single fitness function, the sorting is straight forward. But in many cases, it is necessary to have individuals adapt to multiple fitness functions to represent the desired fitness landscape. Furthermore, one fitness condition might have a higher weight than the others.. It is hard to combine different fitness functions, usually because it is not possible to adjust them to the same scaling, especially if one is considered better with decreasing values. Therefore, competitive ranking was chosen to combine the fitness functions. This means that for each fitness function, the individuals are sorted according to their respective fitness value and a score is assigned to each rank (in the simulator, the score for a rank is the number of individuals that performed worse). Then, for each individual $c_n$ for $m$ fitness functions, the scores $r_i$ multiplied by the respective fitness function weights $w_i$ are summed.

$$\text{fitness score}_{c_n} = \sum_{i=1}^{m} r_i \cdot w_i$$

Finally, the individuals are sorted by their fitness score, which results in their total ranking according to their fitness. Unfortunately, by this procedure, the score loses the information whether the best individual of the creature increased its fitness. However, this can easily be found out by logging all fitness function values of the best creature of the population at the end of the evaluation.

#### 2.1.6.2 Crossover

The typical crossover strategy inspired by biological crossover is commonly known as cutting the two parent genomes into segments and then recombining segments of the parents into a new valid genome. This section additionally describes the procedure of how to choose matching parents.

Using the creatures sorted by the fitness ranking, a percentage of those best performing creatures are chosen to be possible parents. To create new offspring, from every possible

parent a number of offsprings is generated. The number of offspring per parent is found by diving the total number of new creatures to be created by crossover by the number of parents. Having found one of the parent, the other parent is chosen as follows: With $50\,\%$ chance, the parent is chosen as one from a tournament of 10, where the best creature of that tournament is chosen as the second partner. Each tournament creature is drawn from the population using a normal positive integer distribution such that the index of the first creature has the highest probability to be chosen and the $\sigma$ is such that the probability decays nearly completely that the index of the last creature is chosen. With the remaining $50\,\%$ chance, the second parent is chosen at random from the population.

The actual crossover process is very similar to the commonly known one. From the genomes of each parent a subsegment is taken and the second is directly appended to the first one. Unlike the process in classical bit-array notions of genomes is that the two segments have to be interconnected with branches in order to be properly combined. Therefore 10 random branches are mutated such that they interconnect the two subsegments and all gene branches are repaired to reset all invalid branching indices.

#### 2.1.6.3 Grafting

Grafting is similar to crossover in terms of finding two partners (the first called the donor, the second called the receiver), but does not produce offsprings. Instead of just cutting both genomes at random positions and recombining them, it picks a random morphogene in the donor genome and copies this morphogene and its children over to the receiver genome until a certain copying depth $x$ is reached, where $x \in [5, 10]$. As the name says, the idea of this mutation is to copy functional segments to transfer an evolved feature from one creature to another. The functional segments could for instance represent a specific leg or main body structure.

#### 2.1.6.4 Mutations

All other variational operators work on single elements of a single genome. The mutations modify specific aspects of the genome and are specifically made for the genomic structure.

**Gene Specification Mutation** The gene specification mutation applied to a genome iterates over all its morphogene classes. For each class, with probability $0.2$, the morphogene is reinitialized with random values. This affects all values except for the morphogene branches which stay unchanged.

**Gene Split Mutation** The gene split mutation applied to a genome iterates over all morphogene classes. For each class, with probability $0.2$, the morphogene is split along the X, Y or Z axis into two morphogenes occupying roughly the same total volume as the original body part. The morphogenes are then connected with a single branch representing a completely limited joint.

**Gene Purge Mutation** The gene purge mutation applied to a genome purges a gene by mutating it. Then it iterates over all other morphogenes and deactivates all morphogene branches pointing to the gene to be purged. Thereby the purge gene belongs to the non-expressing part of the genome and is therefore purged.

**Branch Specification Mutation** The branch specification mutation applied to a genome iterates over all morphogene classes. For each class, with probability $0.2$, a random morphogene branch is chosen and is reinitialized with random values.

**Grow Stub Gene Mutation** The grow stub gene mutation applied to a genome iterates over all morphogene classes. For each class, with probability $0.6$, $x$ new branches are added to the morphogene, where $x \in [0,2]$. The branches reconnect the stub gene to others and therefore increase the interconnectivity of unconnected morphogene classes.

## 2.2 Evolutionary Cycle

To describe how the different components play together, an example evolutionary cycle of one planet with a flat plane environment and a population of 10 individuals is outlined. One evolutionary cycle can be separated into 2 substeps, the evaluation step and the variation step. Before the cycle can begin, the planet and its environment are set up, the (infinite) flat plane is positioned at the origin of the coordinate system, and each randomly generated individual's genome is transcribed into a phenotype by the Embryogenesis. The population is not yet visible anywhere and awaits evaluation.

### 2.2.1 Evaluation Step

During the evaluation step, each individual is dropped into the simulation slightly above the ground for the selected evaluation period of usually 20 seconds. In single creature mode, which was used for all experiments, every creature is evaluated separately. If a creature consists of zero or only one body part, the creature is discarded, since it is degenerate and

can not move at all, therefore is worthless to be evaluated. Furthermore, the evolution discards creatures that produce very high joint velocities, which is usually an indication that the creature might exploit unrealistic physical forces. During the evaluation of the other creatures, the juries evaluate the performance and calculate the combined fitness value at the end of the evaluation. If all evaluations are over, the evolution switches to the variation step.

### 2.2.2 Variation Step

During the variation step, the reaper picks up the population and culls the worst performing creatures. Leaving a top percentage of creatures without variation (Elitism), it produces offspring by crossover based on the elite and an additional partner, then applies the mutations to the rest of the population. Finally it replaces the remaining number of missing creatures with new, randomly generated creatures.

### 2.2.3 Simulator Specification

The simulator was developed in C++ using a Graphics Engine called Ogre3D [21], a GUI engine called CeGUI [22] and a Physics engine called Bullet Physics [18]. Additionally, the boost library [23] was used to speed up the development of logging data and the creature serialization to disk. The simulator features a rich interface, which can be manipulated using the mouse and the keyboard. Using the graphical interface's menu, a new planet can be configured and instantiated, then a population can be configured and added to it. Furthermore, the gravity of the current planet can be changed to different predefined gravitational values of our solar system. During the evaluation, the evaluation speed can be changed and the simulator can be run in headless mode, meaning that the graphical output is omitted to speed up the simulation. Moreover, the user can move within the simulated space to look at the creature from different angles and can manipulate the creature using the mouse. To observe the current controller and joint dynamics, a window with two graphs can be opened, plotting 50 time steps of the internal states in real-time. If a currently evaluated creature gets stuck in the evaluation or is visibly not fit to be considered, the creature can be culled manually. To look through the evaluated creatures, a population can be loaded and can be sorted out by manually saving some of the creatures and culling others without saving.

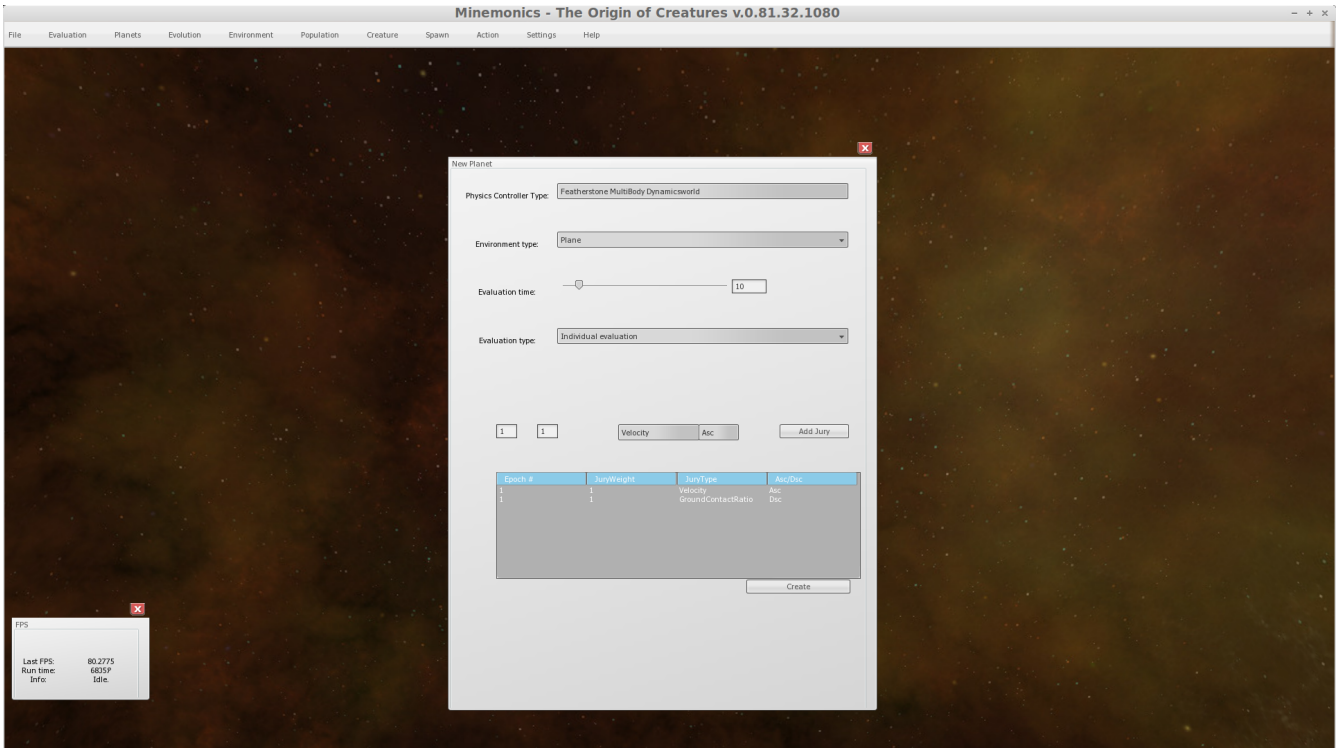


FIGURE 2.12: The figure shows the interface of the simulator when configuring a new planet. In the lower part of the configuration window, the configured juries can be seen.

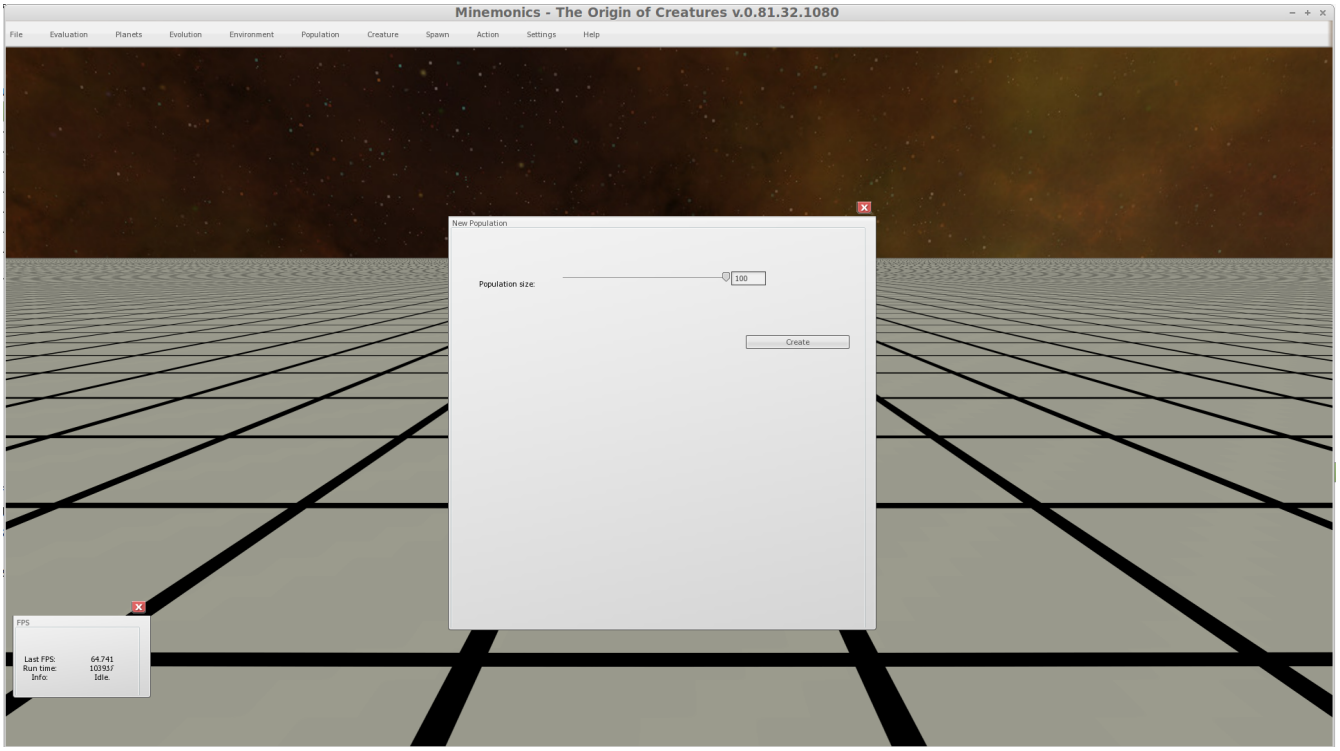


FIGURE 2.13: The figure shows the interface of the simulator featuring when configuring a new population.

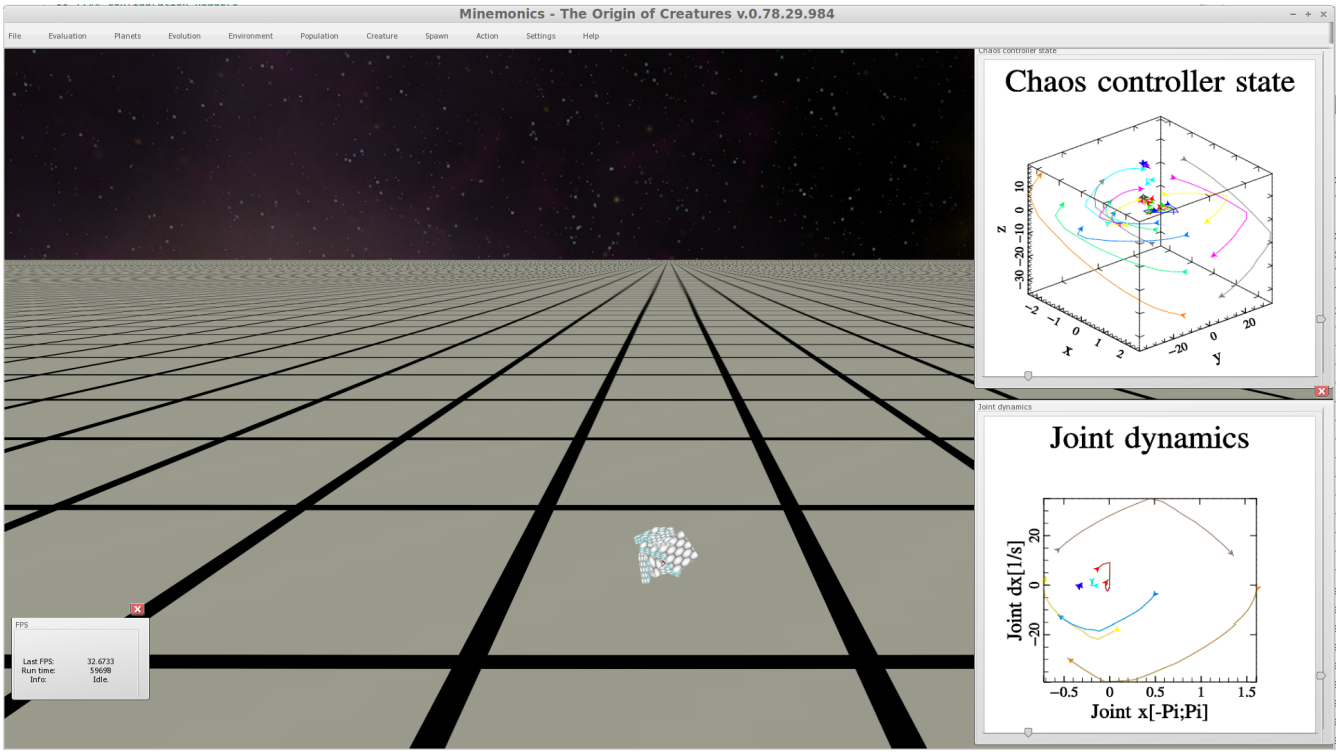


FIGURE 2.14: The figure shows the interface of the simulator featuring a creature in evaluation. On the right-hand side, the window showing the internal graph state can be seen.


FIGURE 2.15: The figure shows the simulator's terminal output providing general information about the running evolutionary process in addition to the more specific, logged information to file.

# Chapter 3

# Controllers

## 3.1 Adaptive Controllers for Locomotion

In this chapter, two different types of controllers, one non-adaptive, sinusoidally oscillating controller and one adaptive, chaotic controller are examined. The non-adaptive, sinusoidal controller is taken to evolve creatures to compare them to the creatures using the adaptive controller. Before we examine the chaotic controller, a short introduction to chaotic systems will be given, then several preliminary experiments on simple limiter control will be conducted to understand the reaction of the controller to different limiter influences. Finally the basic framework of the chaotic controller will be described as it will be used in the creature simulation.

## 3.2 Uncoupled Sinusoidal Controllers

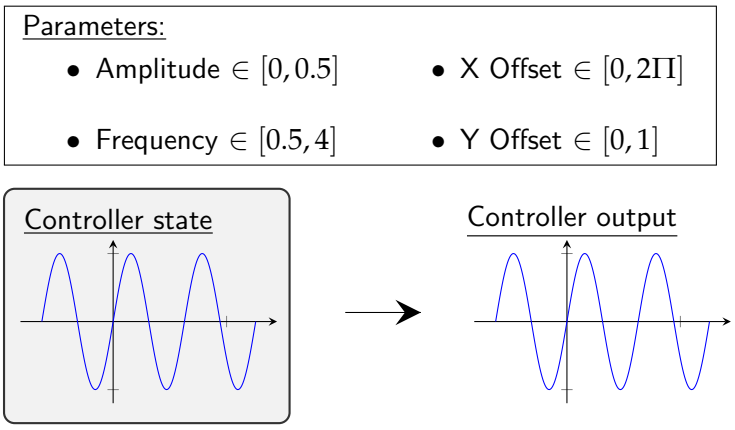


FIGURE 3.1: The specification of the sinusoid controller, its internal state and output signal. The internal state has the same dimensionality as the output. If the sinusoidal controller is mutated during the mutation step of the simulator, the parameters are chosen from a uniform distribution over the respective intervals.

The sinusoidal controller is a simple, input-less controller. Its output is a sinusoidal signal defined by the parameters of amplitude, frequency and the X and Y offset in signal space. The parameter ranges are amplitude $\in [0, 0.5]$, frequency $\in [0.1, 4]$, $\text{offset}_X \in [0, 2\Pi]$ and $\text{offset}_Y \in [0, 1]$. During a mutation of the sinusoidal controller, the amplitude, frequency, X and Y offset are chosen anew from a uniform distribution out of the respective intervals. One creature contains many such controllers, each having its own parameters and each controlling one degree of freedom (DoF) of a certain joint. The controllers are therefore fully distributed within the body of the creature and run completely uncoordinated. Each sinusoidal signal emitted from the sinusoidal controller then is fed into a PID position controller to make the joint DoF follow the sinusoidal signal precisely. In other words, the PID regulates the joint to the position instructed by the sinusoidal controller. Also it must be noted that the PID parameters are hand tuned to make the PID properly follow the given position. A locomotion pattern solution for a creature is therefore a set of sinusoids, that, when applied to the co-evolved morphology, results in a locomotion behavior that moves the creature forward. The controllers are mutated if a Morphogene branch defining a joint is mutated by reinitializing all parameters with random values out of the respective parameter ranges. The controller, however, is non-adaptive within an individual evaluation, therefore the locomotion pattern is non-adaptive as well. This can be seen when the controller state is observed if the creature is on and lifted above ground. The oscillation of the sinusoid stays exactly the same in both cases. Consequently, a solution is only adapted by evolution to the task it was trained for and might not be suitable for a different, even if similar, task.

## 3.3 Chaotic Controllers

The chaotic controller is based on a chaotic, complex, non-linear system (see 3.3.1). The controller has its own internal dynamics and is either completely uninfluenced from outside or is influenced by sensory input from the morphology of the creature, which adapts the controller's dynamics. The internal state of the chaotic controller can be of a higher dimensionality than the output signal. The controller's output is a trajectory of the underlying chaotic system, of which one dimension is then directly applied as a torque on the respective controlled degree of freedom. The output is only scaled by a torque scaling factor, which considers the appropriate torque needed to move the joint DoF to be controlled. The controller's output varies between chaotic and periodic output trajectories depending on how it is influenced. In an uninfluenced state, the controller only exhibits the original system's chaotic trajectory. The trajectory can however be influenced using a chaos control method called simple limiters, which changes the controller's output trajectory from chaotic to different periodic orbits, depending on the respective application of the limiter. We will first introduce chaotic systems which helps to understand the following experiments with simple limiter control on a chaotic system called the Chua circuit [24].

### 3.3.1 Chaotic Systems

A chaotic system is a non-linear, dynamical system exhibiting chaotic dynamics. Chaos means that even though a system is deterministic, therefore also the future behavior is fully determined by the initial state of the system, the future of the system can not be predicted any more if the initial state is slightly varied. This property is one of the following three properties originally described by Robert L. Devaney [25] to characterize chaotic dynamics:

The chaotic system must...

1. be sensitive to the initial conditions.
2. show topological mixing.
3. have dense periodic orbits.

**Sensitivity to the initial conditions**, commonly known as the butterfly effect, means that small differences in the initial conditions lead to widely (usually exponentially) diverging behavior. As Edward Lorenz has put it, the present determines the future, but the approximate present does not approximately determine the same future.

**Topological mixing** can be intuitively seen as in the example of the mixing of dyes, meaning that when the system evolves over time, it causes all regions of the phase space to be mapped

onto each other by the dynamical system. Given any two open sets A and B out of the phase space of a dynamical system defined by the recurrence relation

$$s_{n+1} = f(s_n)$$

, topological mixing means that there is some integer N, such that

$$f^N(A) \cap B \neq 0$$

That means that no matter where the dynamical system starts, eventually the system state will wind up arbitrarily close to any open set.

**Periodic orbits are densely packed** in a chaotic system. This means for a certain point in the phase space that there is always a periodic orbit within any distance $> 0$. Together with the topological mixing, this results in a high degree of unpredictability. Since the infinitely many periodic orbits are unstable, a trajectory might stay on one periodic orbit for a while, then leaves it again and dives through the phase space and suddenly follows another periodic orbit.

Chaotic systems can be defined in a continuous or discrete manner. Continuous systems must be at least 3 dimensional according to the Poincaré-Bendixson theorem [26] in order to be able to show chaotic behavior. Intuitively, this makes sense as no trajectory can pass itself in 2D without touching, which then would lead to a periodic motion because of the deterministic nature of the system. Continuous systems are described using differential equations, which in general can only be solved numerically. Discrete systems can show chaotic behavior also in less than three dimensions, because their representation in difference equations leads to hops in the state space. Consequently, the trajectory does no longer have to explicitly cross another and so does not conflict with determinism. In the following, we will look at a particular continuous chaotic system called the Chua circuit, which is a classical example of chaos, giving rise to the multiscroll attractor.

### 3.3.2 Chua Circuit

According to [27], for a circuit with time-varying output and no time-varying input that is built from electronic components such as resistors, capacitor and inductors, three criteria must hold in order to display chaotic behavior. It must contain:

1. at least one nonlinear element (where piecewise linear is sufficient)
2. at least one locally active resistor
3. at least three energy-storage elements

The most basic electronic circuit that satisfies these criteria is the Chua circuit [24]. Chua's circuit is chaotic system that can be built in the form of a simple electronic circuit. The Chua circuit was invented by Leon O. Chua when he visited the Waseda University in Japan in 1983. The circuit can be seen as a non-periodic oscillator, that, opposed to an ordinary electronic oscillator, never repeats its waveform. The circuit, remarkable because of its simplicity and rich variety of bifurcations and the presence of chaotic behavior, is one of the few physical systems for which the presence of chaos has been proven mathematically [27].

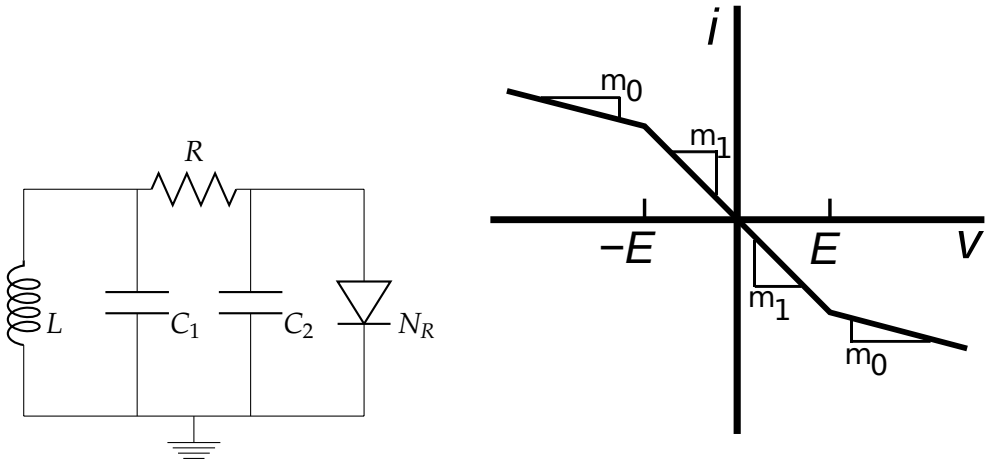


FIGURE 3.2: The Chua circuit with its special Chua diode. The Chua diode is a piecewise-linear resistor with the characteristics as shown in the right figure.

Using Kirchhoff's circuit laws to derive the equations of the Chua circuit, we find the following non-linear ordinary differential equations with the variables x(t), y(t) and z(t).

$$\frac{dx}{dt} = \alpha(y - x - f(x)) \qquad \frac{dx}{dt} \text{ is the voltage across the capacitor } C_1$$

$$RC_2\frac{dy}{dt} = (x - y + Rz) \qquad \frac{dy}{dt} \text{ is the voltage across the capacitor } C_2$$

$$\frac{dy}{dt} = \beta(x - y + Rz) \qquad \text{with } \frac{1}{RC_2} \text{ being } \beta$$

$$\frac{dz}{dt} = -\gamma y \qquad \frac{dz}{dt} \text{ is the current across the inductor } L_1$$

$$f(x) = \frac{m_1 x + (m_0 - m_1)}{2(|x + E| - |x - E|)} \qquad f(x) \text{ describes the response of the piecewise linear resistor}$$

The Chua circuit was chosen as a model system of a chaotic controller because of its simple definition through the equations as well as it being a prior example system for simple limiter control in [2]. Corron et al. showed in two different chaotic systems, namely in the driven chaotic pendulum and the Chua circuit, how appropriate simple limiters can be applied to each system so that the chaotic behavior of each unbounded system could be controlled into behaviors of different periodicities. Important to mention is that the Chua Circuit is not

meant to be a model for appropriate leg movement. The goal is to show that using simple limiters to control an arbitrary chaotic system, the different periodicities generated by the system leading to periodic leg movement could form different locomotion gaits. The change of the terrain representing a change of the simple limiters can lead to a change in periodic movement, thereby adaption to different environments occurs.

#### 3.3.2.1 Simple Limiter Control in Mathematica

Before the circuit was used as a chaotic controller, it was modelled in Mathematica to observe its original chaotic behavior and influence it by simple limiters to exhibit different periodicities. Two experiments were conducted to look at two different limiter configurations in detail. One experiment explores the features of a self-limiting dimension and a dimension limiting another dimension. Another experiment reveals the influence of softness of the limiter on the control behavior in both of them. The differential equations were integrated using a Runge-Kutta integration scheme of order 6 with an integration step of $h = 0.001$. The constants were set as $\alpha = 15.6$, $\beta = 1$, $\gamma = 28$, $m_0 = -0.714$, $m_1 = -1.143$ and $E = 1$. The figure 3.3 shows the Chua circuit's multiscroll attractor using initial conditions $(x(0), y(0), z(0)) = (-1.5, 0, 0)$ without any limiter. These initial conditions are used during the following experiments unless otherwise stated.

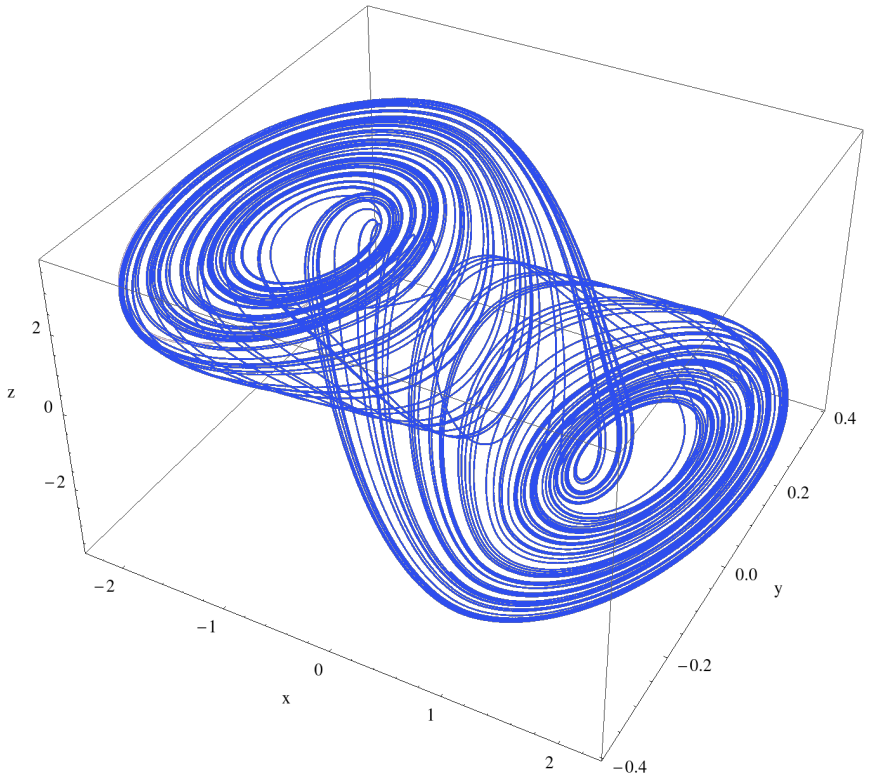


FIGURE 3.3: The multiscroll attractor generated by the Chua circuit without any simple limiter applied.

Then we introduce a simple tunable soft limiter

$$\text{softLim}(x) = \frac{1}{2}\left(\tanh\left(\frac{limitValue - x}{softness}\right) + 1\right)$$

with the limiter response shown in figure 3.4. The limiter function is applied to one of the differential equations of the chaotic system and pulls its result to zero as it or one of the other equations surpass the $limitValue$ depending on which equation is defined to be the one triggering the limiter and which is the one to be limited. In all of the following plots, the color of a certain state in the plot indicates how strongly the limiter influences that state.

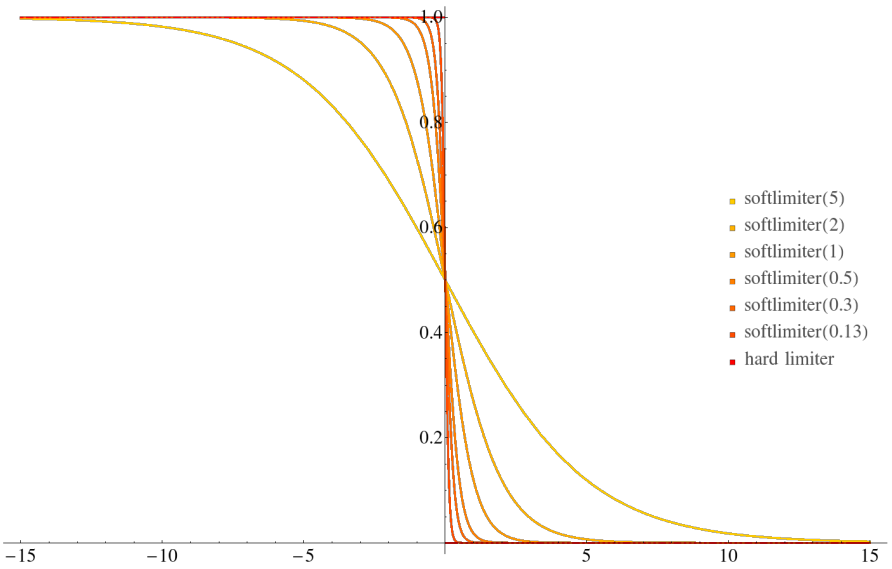


FIGURE 3.4: Soft limiter responses compared to the hard limiter response. The limiter function is applied to one of the differential equations of the chaotic system and pulls its result to zero as it or one of the others surpass the $limitValue$ depending on which equation is defined to be the one triggering the limiter and which is the one to be limited. The hard limiter response is shown in red. A trajectory approaching 0 from the negative side runs into the limiter, which applies the limiter instantly to its full extent. The soft limiter responses from orange to yellow show how limiters with increasing softness values (0.13, 0.3, 0.5, 1, 2 and 5) are applied. The curve with softness 0.13 approximates the hard limiter the best, the higher the softness value, the softer the limiter.

Initially a hard limiter defined as a piecewise function was chosen, however a soft limiter is a better model for physical limiters, because in reality no ideal hard limiters exist. Using the $softness$ parameter of the limiter, the softness can be tuned, where a $softness = 10$ means very soft and a $softness = 0.1$ signifies a very hard limiter. The $limitValue$ parameter defines the threshold value, at which the limiter is applied exactly half when surpassed. With the above mentioned initial conditions, the trajectory in state space stays within the bounding box of

$$x \in [-2.8717, 2.2417]$$
$$y \in [-0.380467, 0.386978]$$
$$z \in [-3.59156, 3.62959]$$

**One Dimension Limiting Another** In the first experiment, the softness was set to $softness = 0.13$ to approximate a rather hard limiter. The limiter is added to the equations as shown below:

$$\begin{aligned}
\frac{dx}{dt} &= \alpha(y - x - f(x)) \\
\frac{dy}{dt} &= \beta(x - y + Rz) \\
\frac{dz}{dt} &= -\gamma\ y\ \text{softLim}(x) \\
f(x) &= \frac{m_1 x + (m_0 - m_1)}{2(|x + E| - |x - E|)}
\end{aligned}$$

If the $limitValue$ parameter is changed from $> 3.19$, which corresponds to no limiter influence to the trajectory, to a value within $limitValue \in [2.28, 3.19]$, different chaotic trajectories can be observed. The limiter influences the circuit by limiting the change of current flow $\frac{dz}{dt}$ when $x$ approaches the limiter.

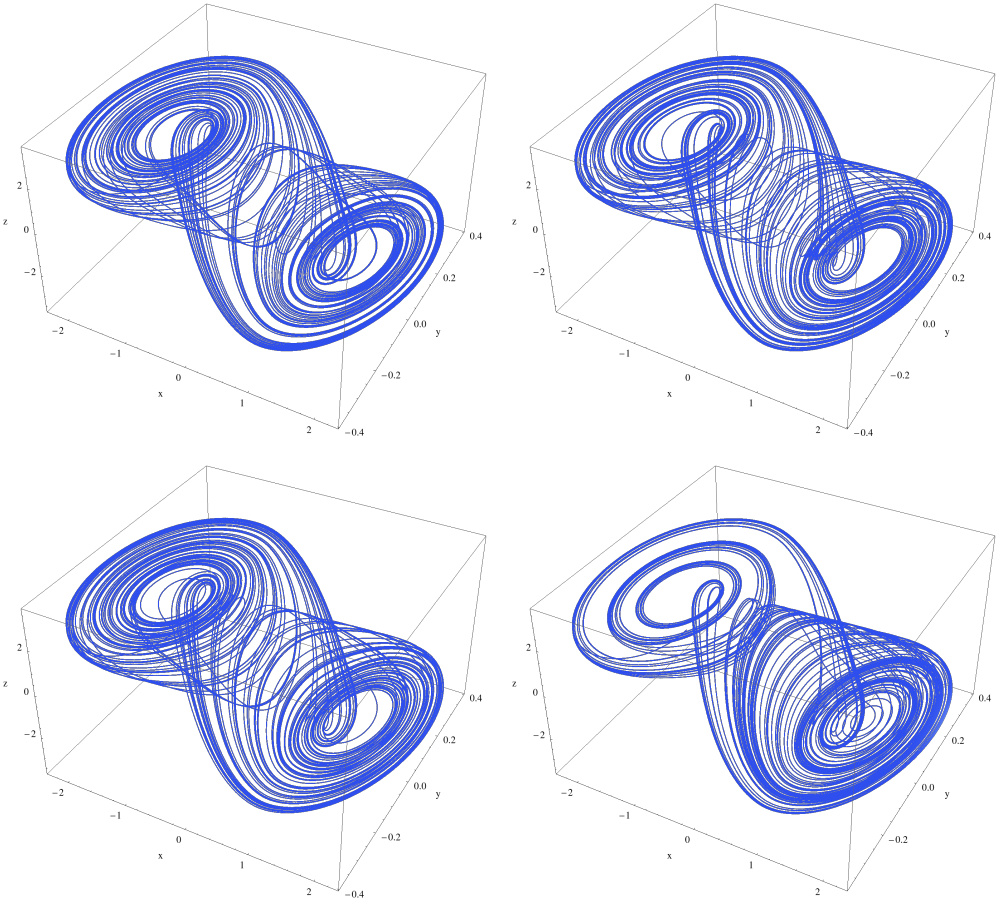

FIGURE 3.5: Figure of chaotic behaviors in range 2.4-3.19. The limit values of the shown figures are 3, 2.8, 2.6 and 2.4.

When choosing a $limitValue \in [2.28, 2.4[$, one can observe that the limiter more and more suppresses chaotic behavior and reduces the number of times the trajectory switches back into the uppermost scroll of the multiscroll.

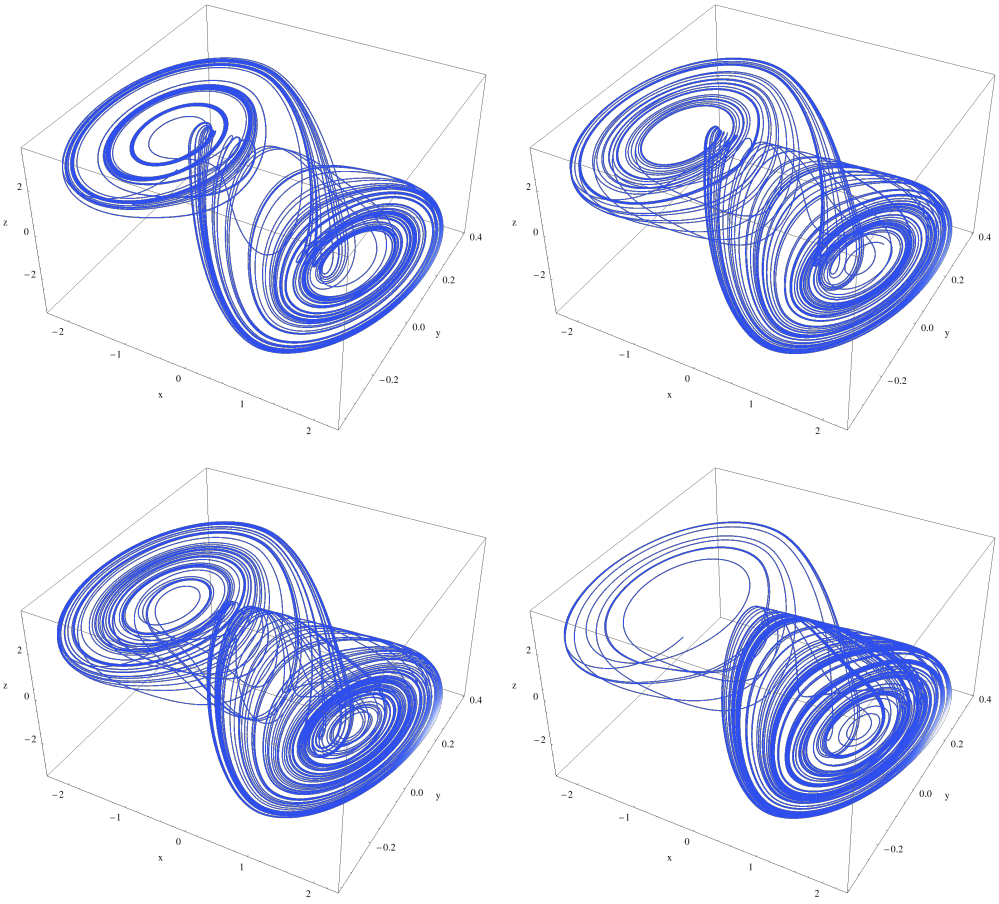

FIGURE 3.6: Figure of behaviors in range 2.28-2.4. The limit values of the shown figures are 2.375, 2.35125, 2.30375 and 2.28. In the lower left and right plot, the slightly white coloring of the curve indicates the strength of the limiter influence.

With a value of $limitValue \in [1.81, 2.28]$, the state only stays in the lowermost scroll and shows periodic behavior. The control behavior can be explained by looking at figure 3.3 again. The trajectories that lead from the lower-most scroll to the upper-most scroll pass by at the outer orbits of the lower-most scroll and therefore have the highest x(t) values. The limiter is applied when a certain x(t) value is surpassed, therefore the limiter limits z(t), which then leads to a repelling influence from the limiter. The trajectory thereby is pulled back onto a former orbit, which stabilizes the state onto that orbit. However, since the limiter is a very hard limiter, the period of the trajectory shows higher periods only in a very small limit value parameter window $[2.212, 2.215]$ and then very quickly turns into period 1 within the parameter window $[1.83, 2.212]$ with quicker convergence the nearer the limiter was placed to zero. The system is highly sensitive to the exact position of the limiter because of the dense periodic orbits in the system, between which the state quickly switches when the limiter is slightly moved. One can see the convergence to the limit cycle vary, getting slower again with $limitValue \in [2.183, 2.15]$ increasing again for $limitValue \in [2.15, 1.83]$. A self-similar pattern of increasing and decreasing convergence speeds can be observed.

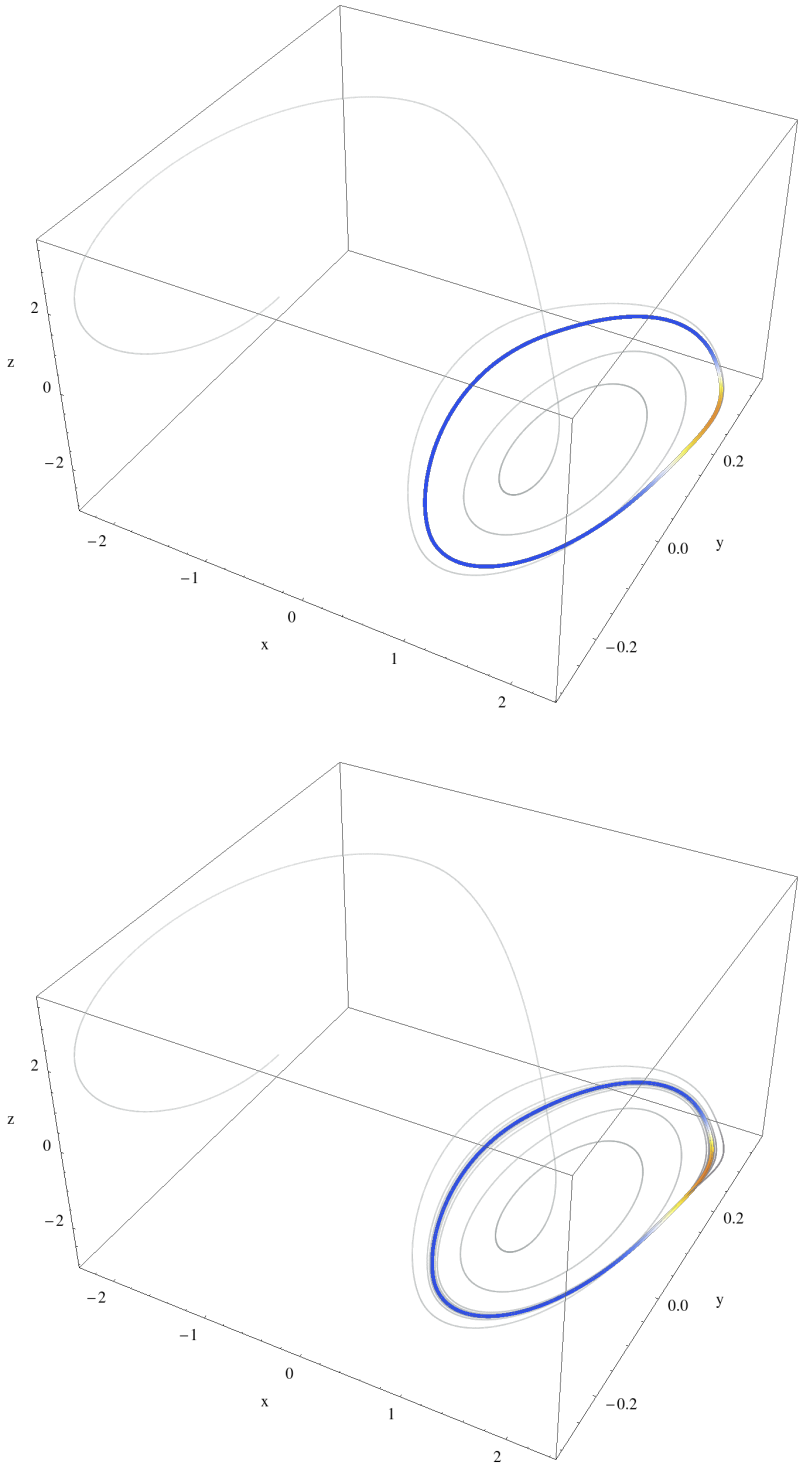


FIGURE 3.7: Figures of period 1 limit cycle with one with a fast and one with a slow convergence to the limit cycle. The limit values are 2.183 for the upper and 2.15 for the lower plots. The convergence to the limit cycle is grayed out so that the final limit cycle's visibility is enhanced.

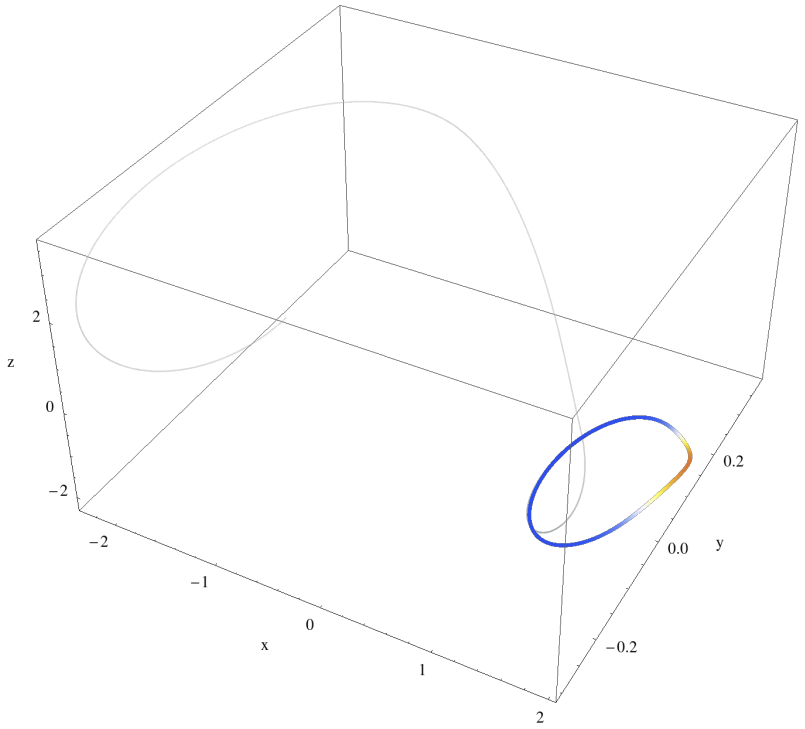


FIGURE 3.8: Figure of period 1 limit cycle with again a fast convergence to the limit cycle using a limit value of 1.83. The plot to the right does not show the convergence trajectory, but instead show the periodic orbit after proper convergence.

Below 1.81, the limiter is applied instantly, the trajectory drops in z direction to $\approx -2.68343$ and then heads towards $[\infty, \infty, -2.68343]$, therefore can never escape the limiter again.

The limiter control gets more interesting when the softness is set to $softness = 0.5$, which corresponds to a softer limiter (Fig. 3.4). The softer limiter is applied earlier than the hard limiter because of its long tails from the $\tanh()$ function, however, its influence is rather minimal yet gradually increasing when the limiter is approached. An extended parameter window for the limiter parameter can now be found, being $limitValue \in [2.3887, 11.5065]$, where the limiter suppresses chaotic behavior and again reduces the number of times the trajectory switches back into the uppermost scroll.

Within the parameter window $limitValue \in [2.05888, 2.387[$, the influence of the limiter keeps the trajectory in the lowermost scroll. With the softer limiter, it is now much easier to find the parameter window to see the period doubling, because the parameter windows for the different periodicities are now much larger.

The table 3.1 shows different limit values that stabilize different periodicities.

| *limitValue* | Limit Cycle |
|---|---|
| 2.292 | Period 8 |
| 2.285 | Period 4 |
| 2.25 | Period 2 |
| 2.12 | Period 1 |

TABLE 3.1: Different limit values resulting in trajectories of different periodicity where x(t) limits z(t) using a 0.5 soft limiter.

The plots below show the periodic trajectories for various limit values. The convergence trajectory is grayed out again in the following plots.

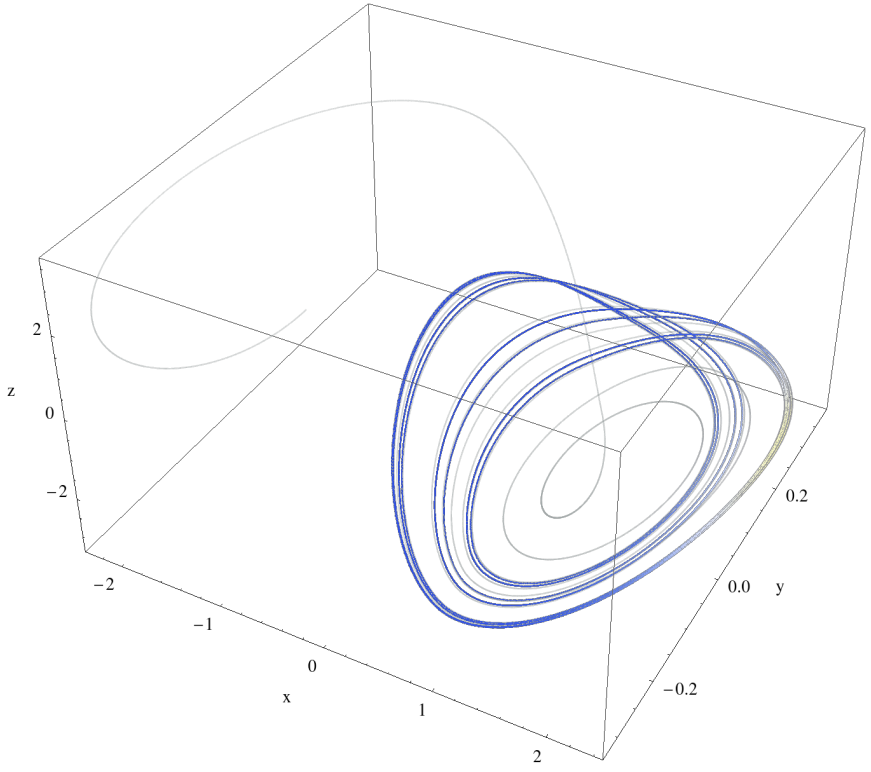


FIGURE 3.9: Figure of period 8 limit cycle where x(t) limits z(t) using a 0.5 soft limiter.

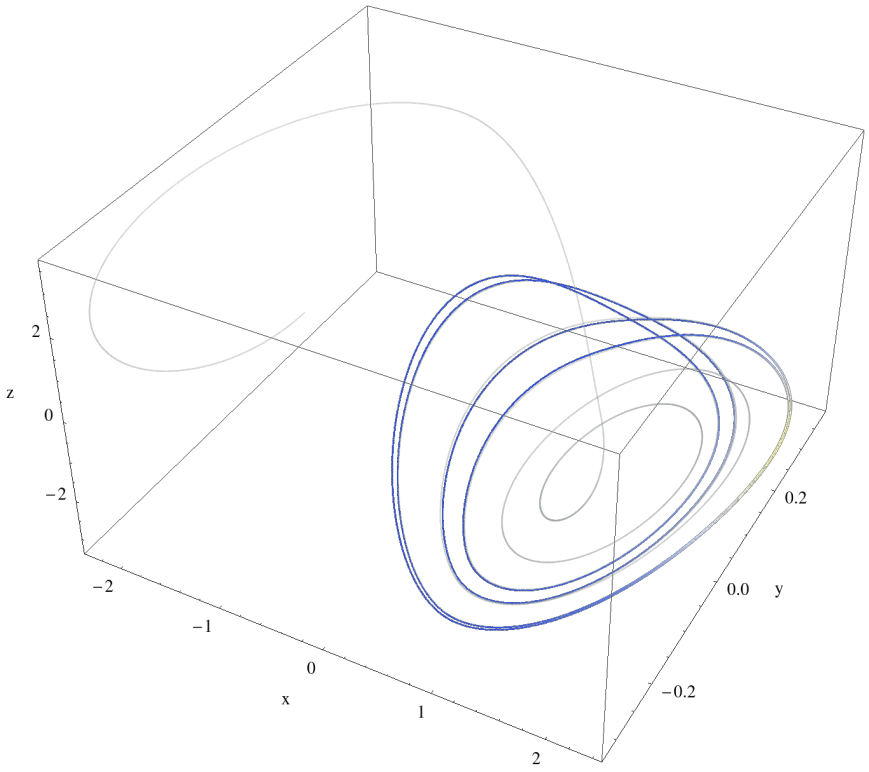


FIGURE 3.10: Figure of period 4 limit cycle where x(t) limits z(t) using a 0.5 soft limiter.

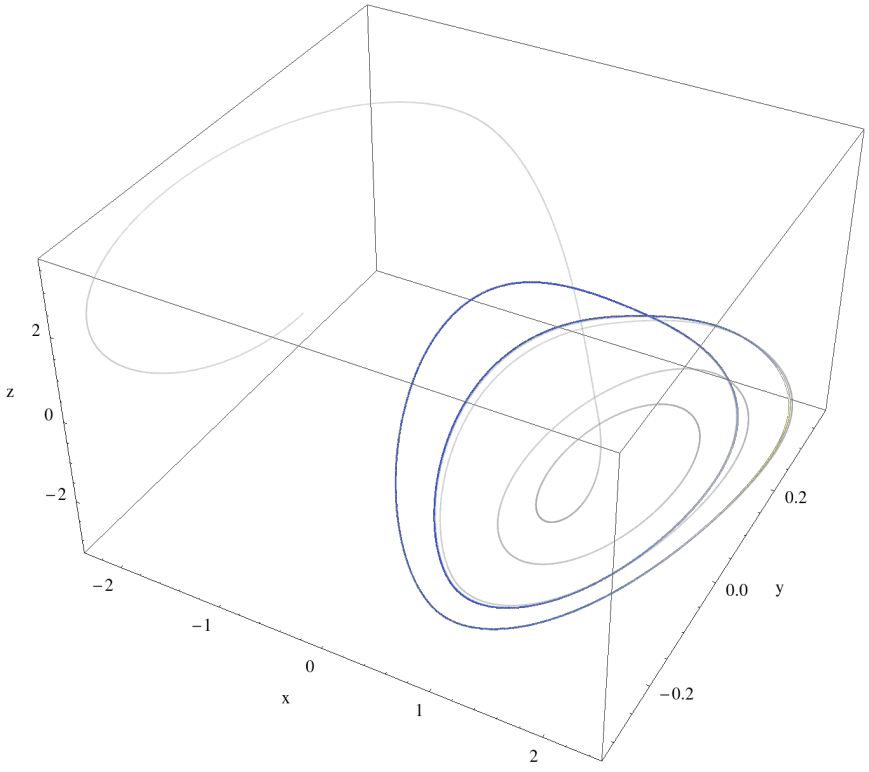


FIGURE 3.11: Figure of period 2 limit cycle where x(t) limits z(t) using a 0.5 soft limiter.

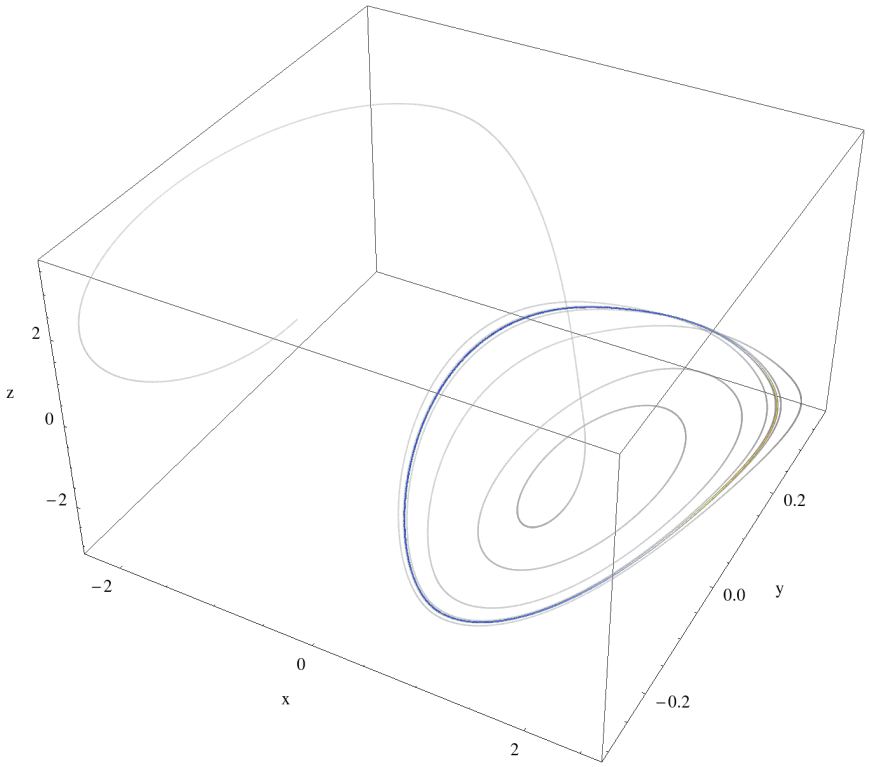


FIGURE 3.12: Figure of period 1 limit cycle where x(t) limits z(t) using a 0.5 soft limiter.

**One Dimension Limiting Itself** In the second experiment, the softnss is again set to $softness = 0.13$ to approximate a rather hard limiter. The limiter is added to the equations as shown below:

$$\begin{aligned}
\frac{dx}{dt} &= \alpha(y - x - f(x)) \text{ softLim}(x) \\
\frac{dy}{dt} &= \beta(x - y + Rz) \\
\frac{dz}{dt} &= -\gamma\, y \\
f(x) &= \frac{m_1 x + (m_0 - m_1)}{2(|x + E| - |x - E|)}
\end{aligned}$$

If the $limitValue$ parameter is changed from $> 4.6692$, which corresponds to no limiter influence to the trajectory, to a value within $limitValue \in [1.99545, 4.6692[$, one can observe different chaotic trajectories. The limiter influences the circuit by limiting the voltage across the capacitor $C_1$ when $x$ approaches the limiter. Within the window $[1.89, 1.99545]$ a small limiter parameter window can be found which controls trajectories of high periodicity down to different lower periods. Period doubling behavior can be found when controlling the Chua circuit appropriately with limit values as described in the table 3.3.

| *limitValue* | Limit Cycle |
|---|---|
| 1.932 | Period 8 |
| 1.93 | Period 4 |
| 1.91 | Period 2 |
| 1.89 | Period 1 |

TABLE 3.2: Different limit values resulting in trajectories of different periodicity, where x(t) limits itself using a 0.13 soft limiter.

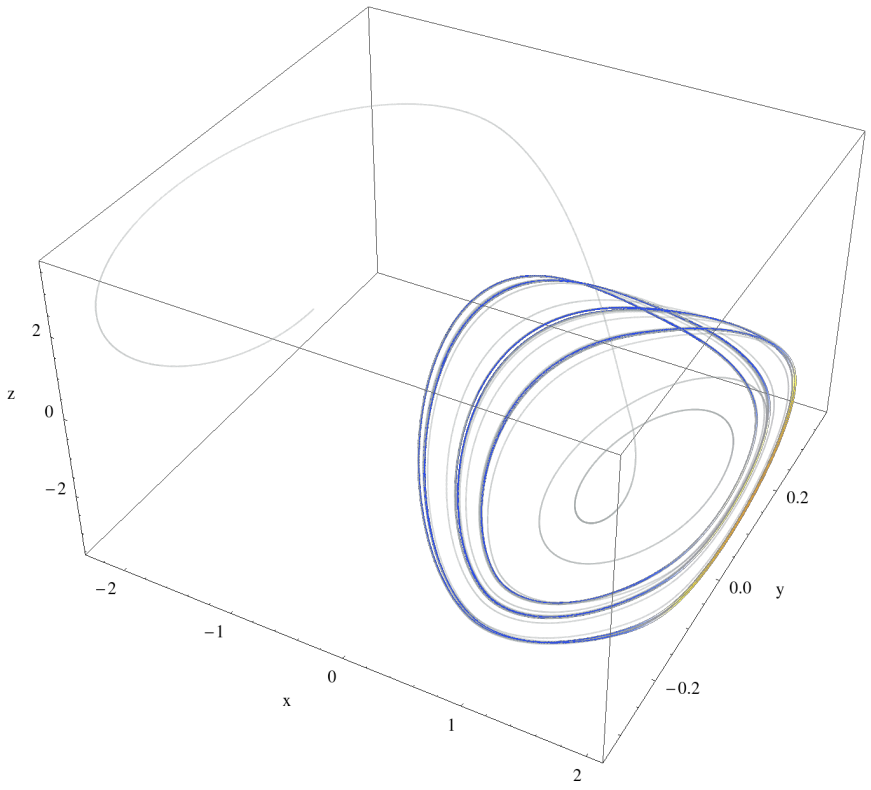


FIGURE 3.13: Figure of period 8 limit cycle when using a self limiter with value 0.13.

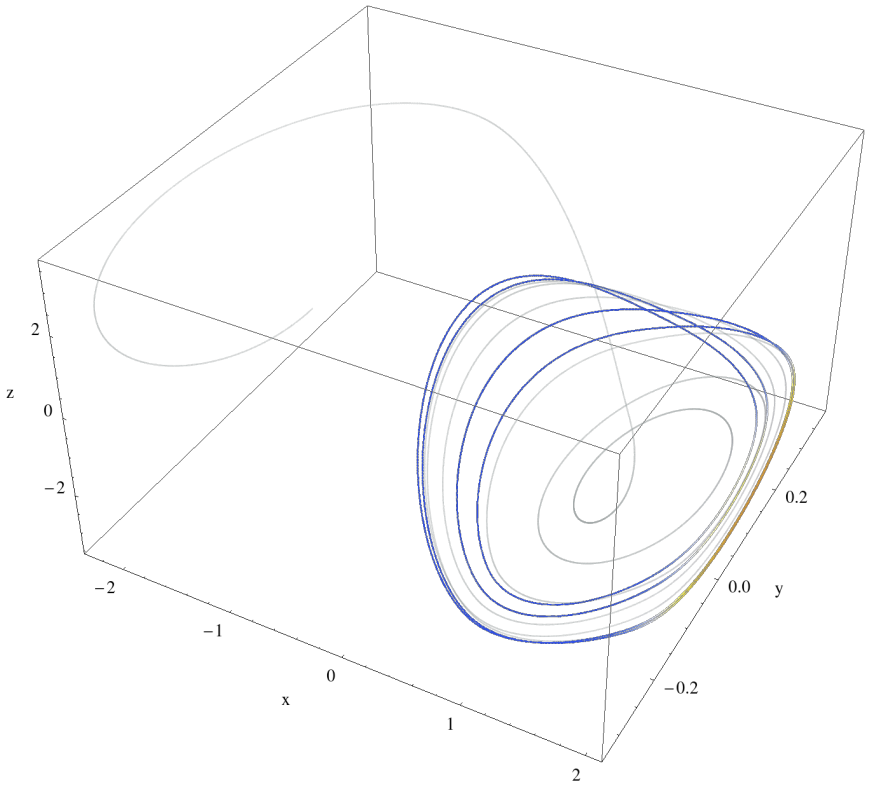


FIGURE 3.14: Figure of period 4 limit cycle when using a self limiter with value 0.13.

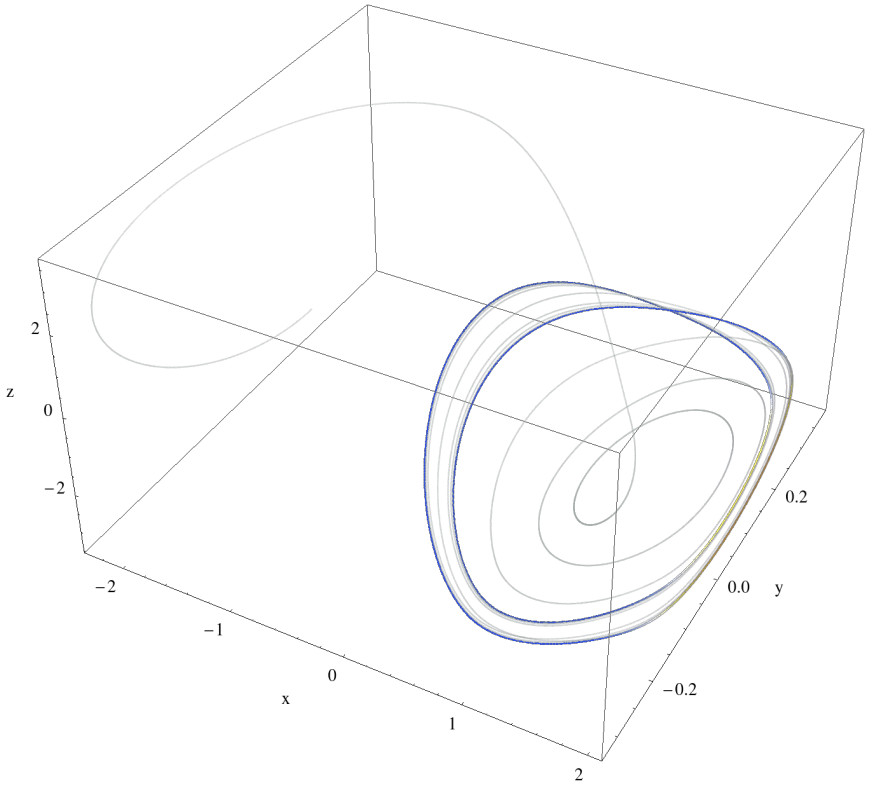


FIGURE 3.15: Figure of period 2 limit cycle when using a self limiter with value 0.13.

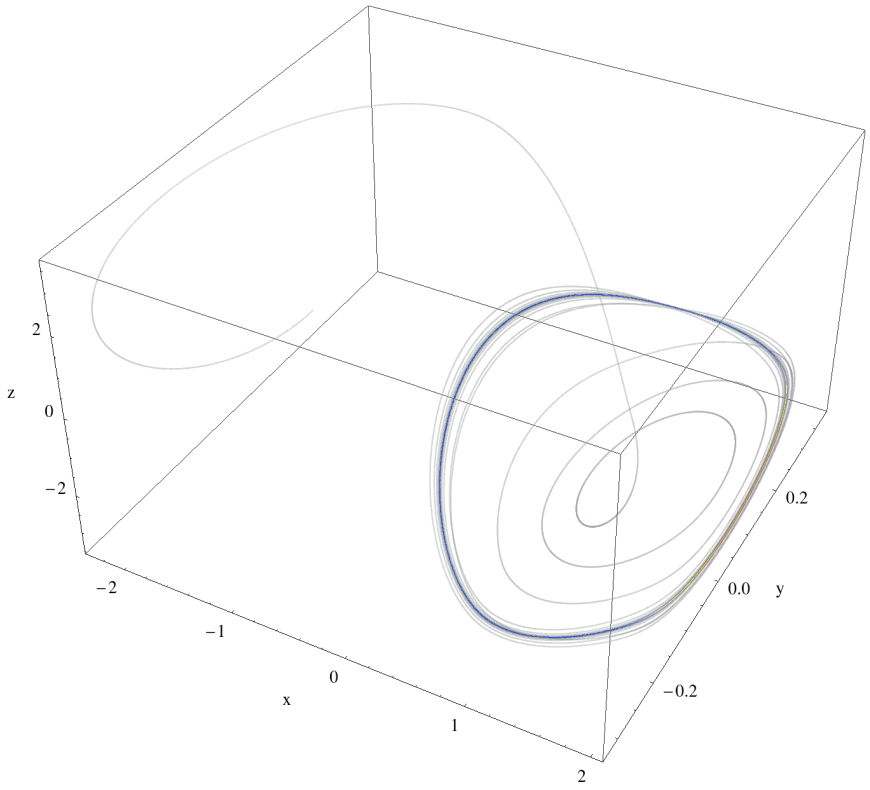


FIGURE 3.16: Figure of period 1 limit cycle when using a self limiter with value 0.13.

The hard limiter leads to a very slow convergence and therefore it is hard to determine the final periodicity because every evaluation takes a lot of time. However, with x limiting itself, the limiter parameter of the system can now be set far beyond the fixed point of the lowermost scroll without the trajectory getting trapped. At $limitValue = 1.61$, one can observe perfect convergence to period 1.

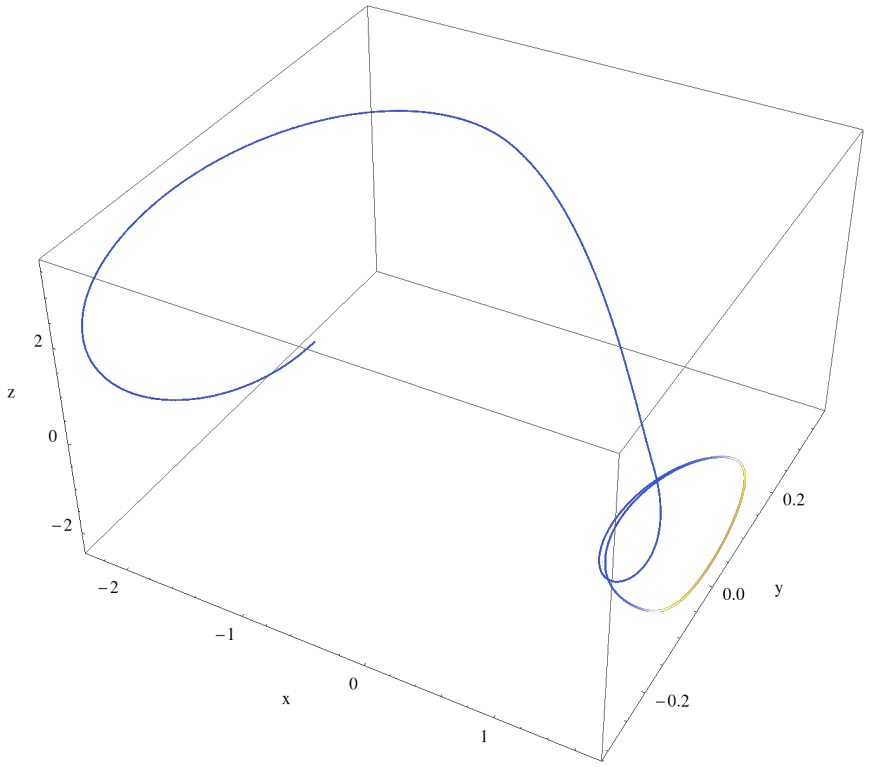


FIGURE 3.17: Figure of period 1 limit cycle at 1.61.

Surpassing this point increases the convergence time again because the trajectory converges to the limit cycle from outside.

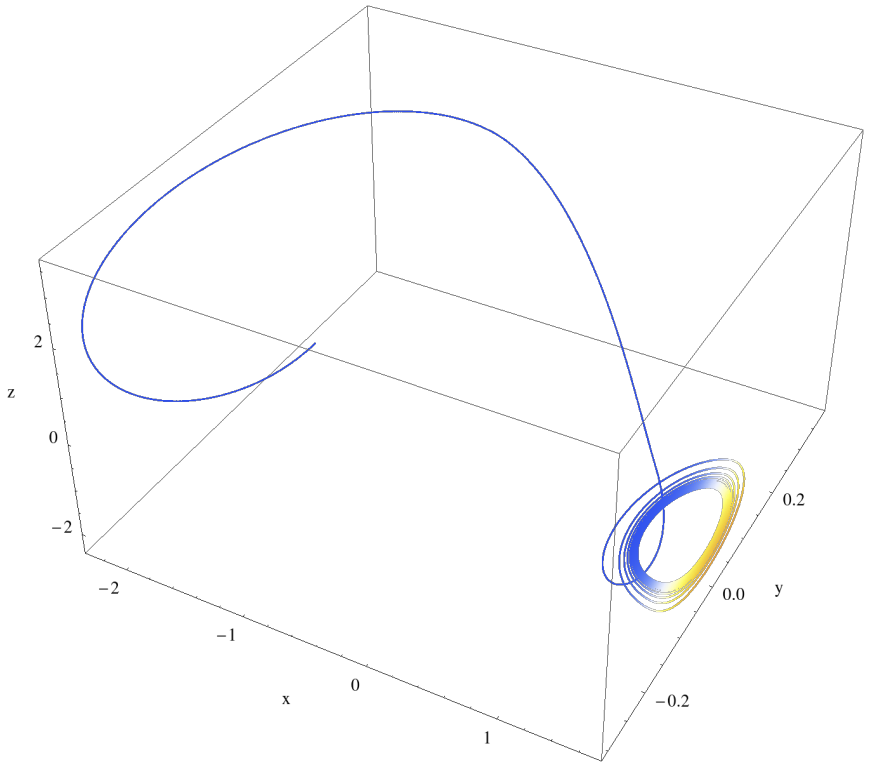


FIGURE 3.18: Figure of period 1 limit cycle at 1.57.

The controlled limit cycle shrinks, until the trajectory directly converges onto one single fix point at $limitValue = 1.4$.

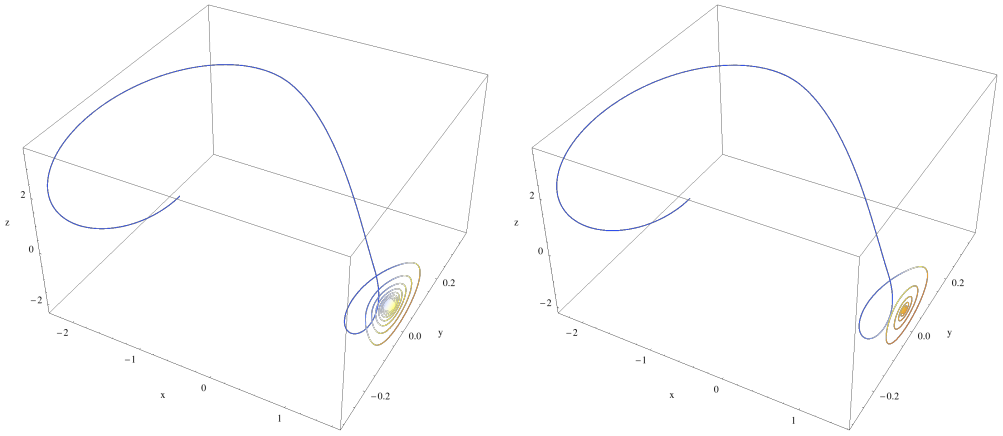

FIGURE 3.19: Figure of trajectories with limit values 1.5 and 1.4.

Moving the limiter to 1.3 leads to a rotation of the plane in which the original trajectories took place, and an inward spiral starting at the initial conditions and progressing towards the limiter occurs. The state spiralling into the limiter tries to pass the limiter from the uppermost the gateway scroll to the lowermost scroll.

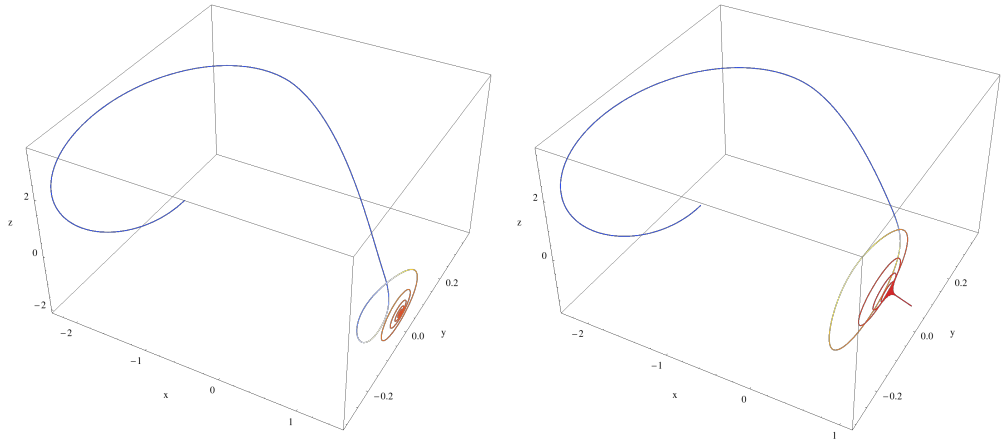


FIGURE 3.20: Figures of trajectories with limit values 1.3 and 0.5.

The spiralling behavior continues until $limitValue = 0.26991$, where the trajectory stays in the uppermost scroll of the multiscroll attractor before it spirals towards the limiter.

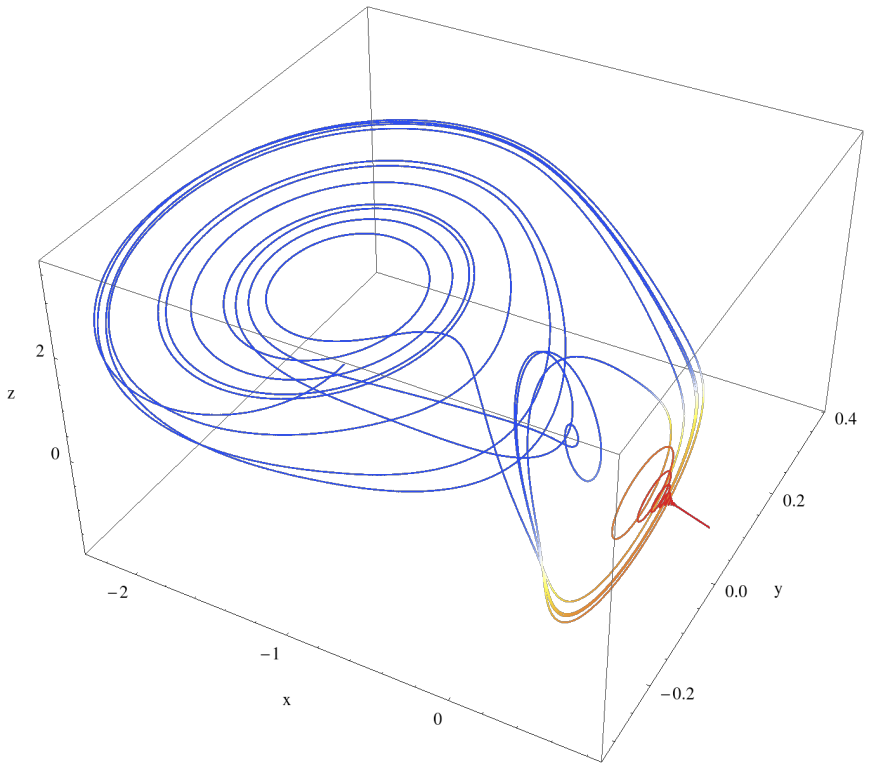


FIGURE 3.21: Figure the trajectory with limit value 0.26991. The state stays in the uppermost scroll before it spirals into the limiter.

In the uppermost scroll, starting at $limitValue = 0.25304$, one can observe high periodic behavior which can be controlled to different periodicities by moving the limiter towards the scroll.

| *limitValue* | Limit Cycle |
|---|---|
| -0.126 | Period 32 |
| -0.127 | Period 16 |
| -0.1327 | Period 8 |
| -0.15 | Period 4 |
| -0.2 | Period 2 |
| -0.34 | Period 1 |

TABLE 3.3: Different limit values resulting in trajectories of different periodicity in the uppermost scroll.

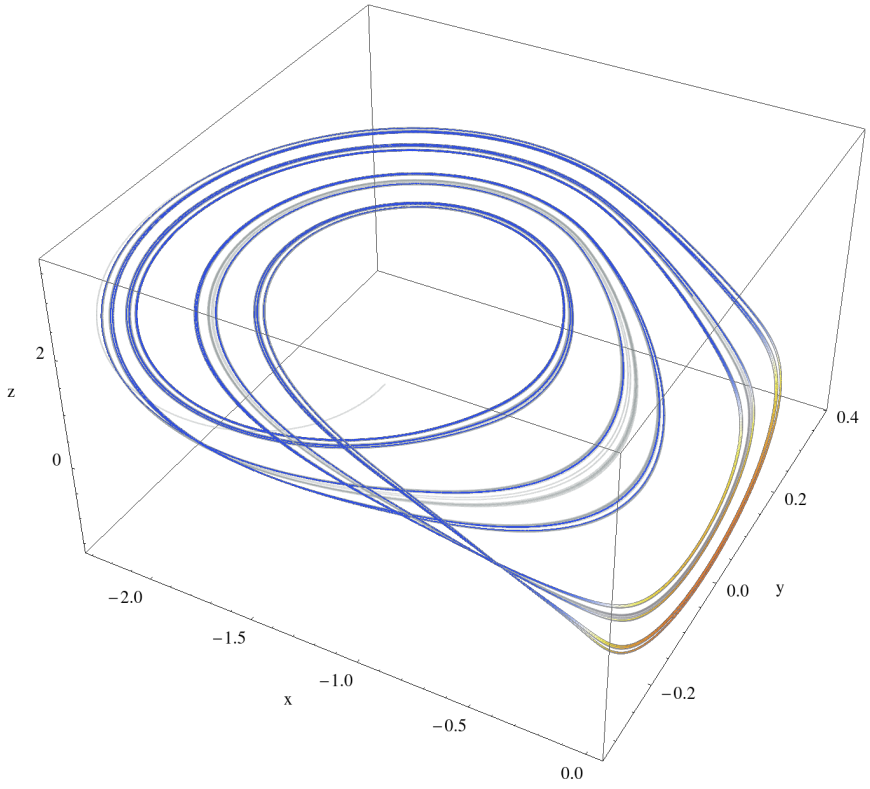


FIGURE 3.22: Figure of period 32 limit cycle in the upper scroll using a self limiter with value 0.13.

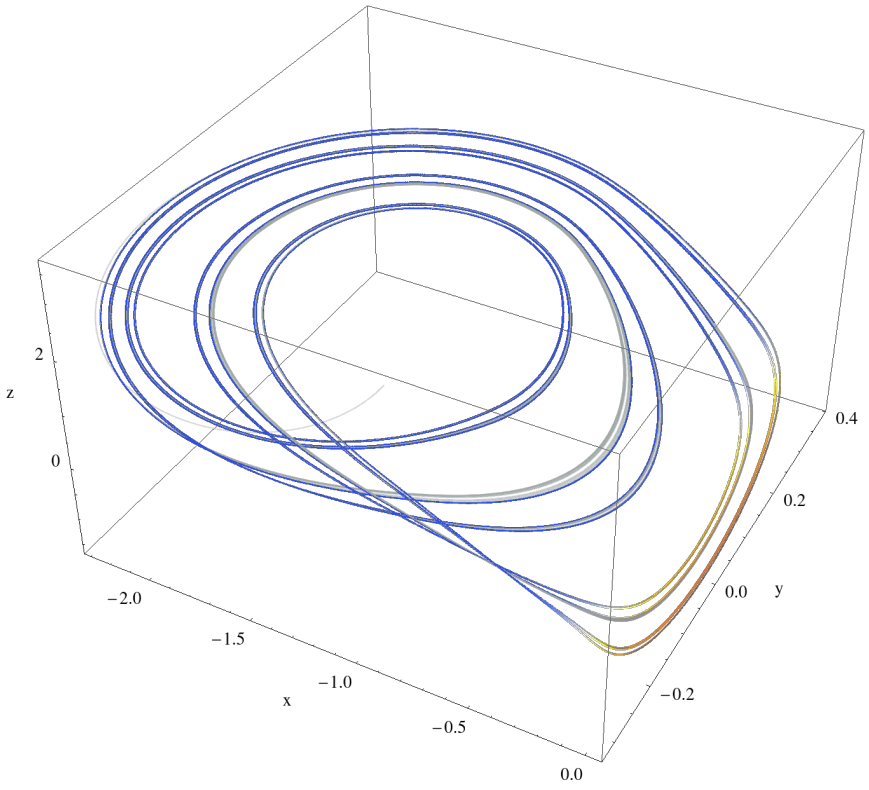


FIGURE 3.23: Figure of period 16 limit cycle in the upper scroll using a self limiter with value 0.13.

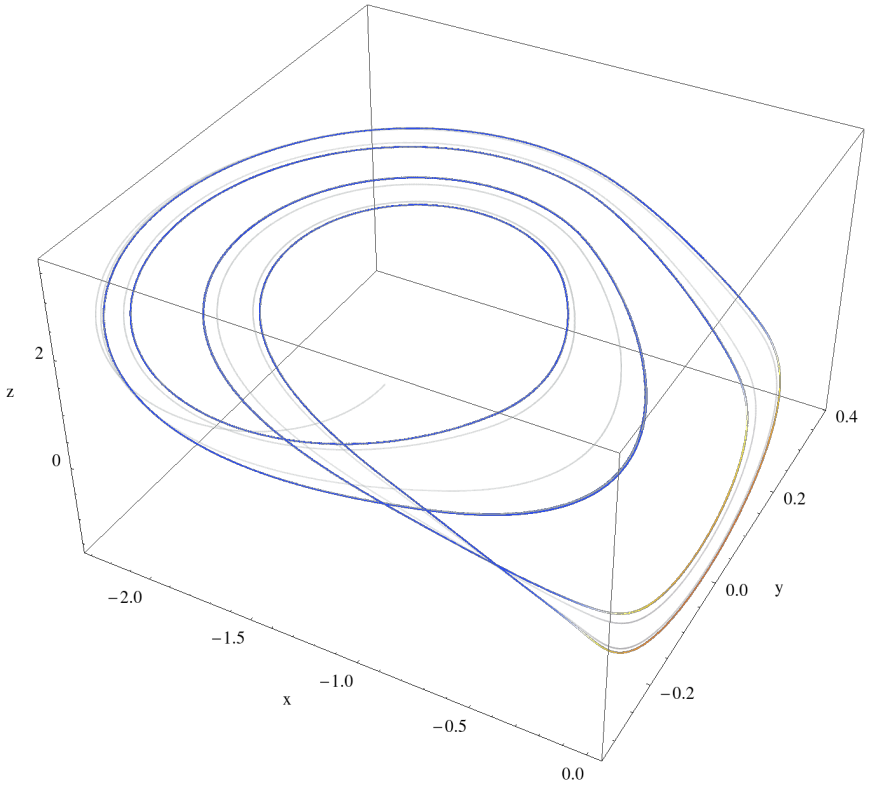


FIGURE 3.24: Figure of period 8 limit cycle in the upper scroll using a self limiter with value 0.13.

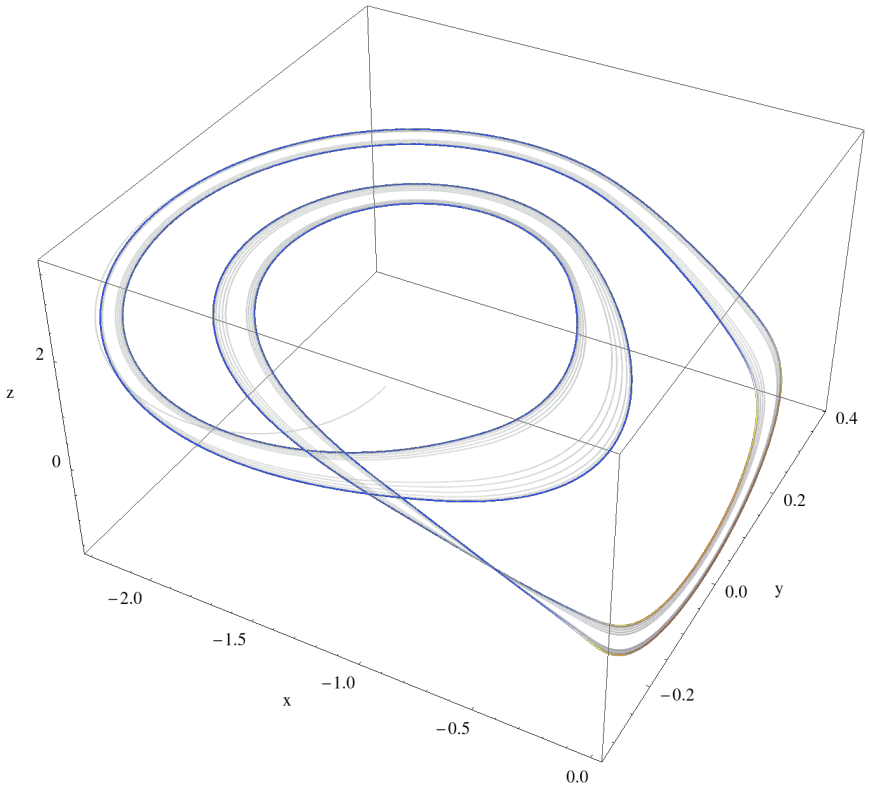


FIGURE 3.25: Figure of period 4 limit cycle in the upper scroll using a self limiter with value 0.13.

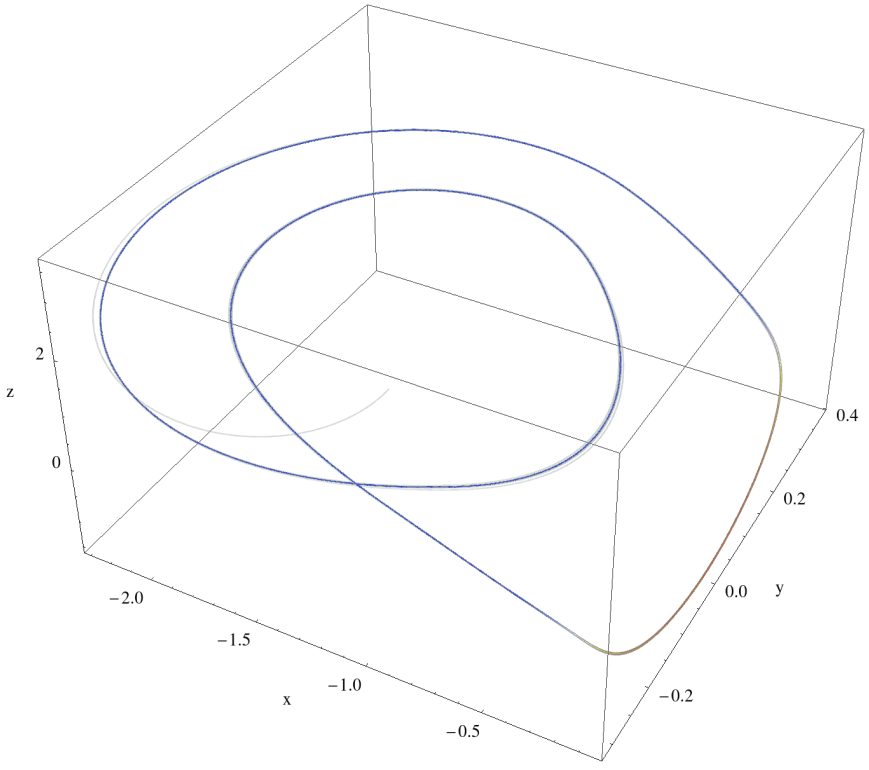


FIGURE 3.26: Figure of period 2 limit cycle in the upper scroll using a self limiter with value 0.13.

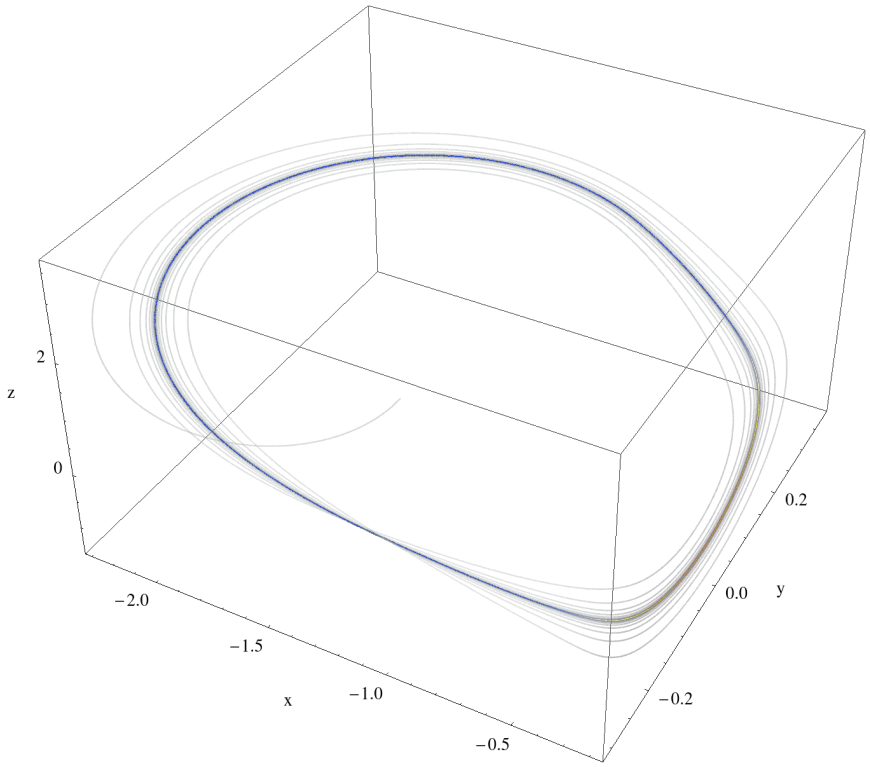


FIGURE 3.27: Figure of period 1 limit cycle in the uppermost scroll.

Using $softness = 0.5$, the limiter starts to influence the multiscroll attractor at $11.507$, and the chaotic behavior can again be controlled using the limiter within the limiter value window $[2.176, 11.507]$. Within the window $[1.8, 2.176[$, periodic orbits down to period 1 can be stabilized.

| *limitValue* | Limit Cycle |
|---|---|
| 2.0502 | Period 16 |
| 2.05 | Period 8 |
| 2.04 | Period 4 |
| 2 | Period 2 |
| 1.8 | Period 1 |

TABLE 3.4: Different limit values resulting in trajectories of different periodicity

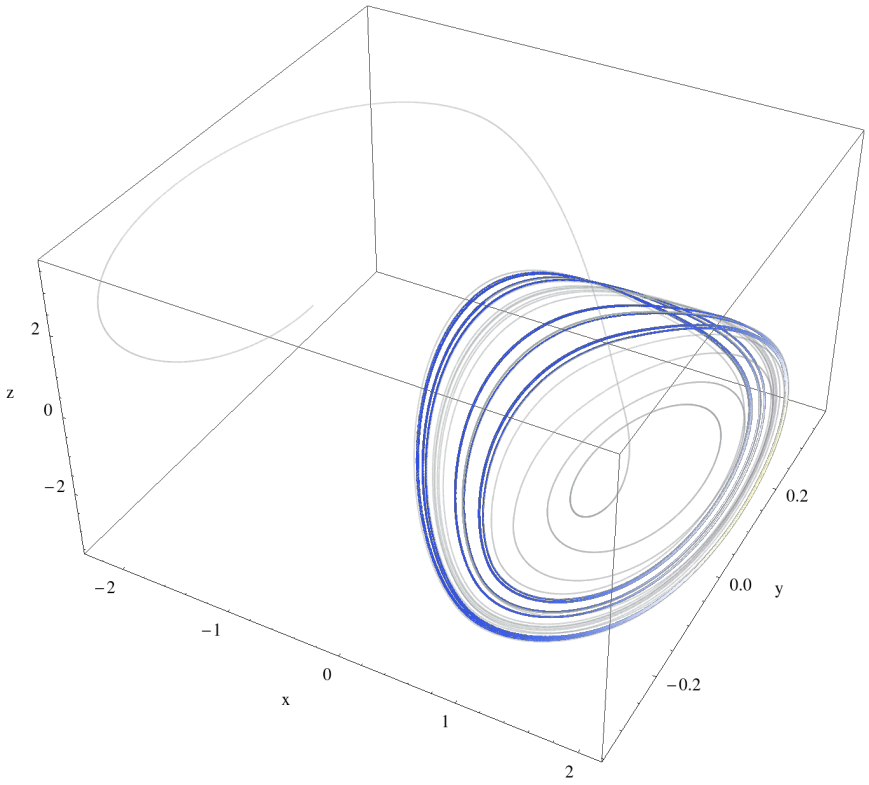


FIGURE 3.28: Figure of period 16 limit cycle using a self limiter with value 0.5.

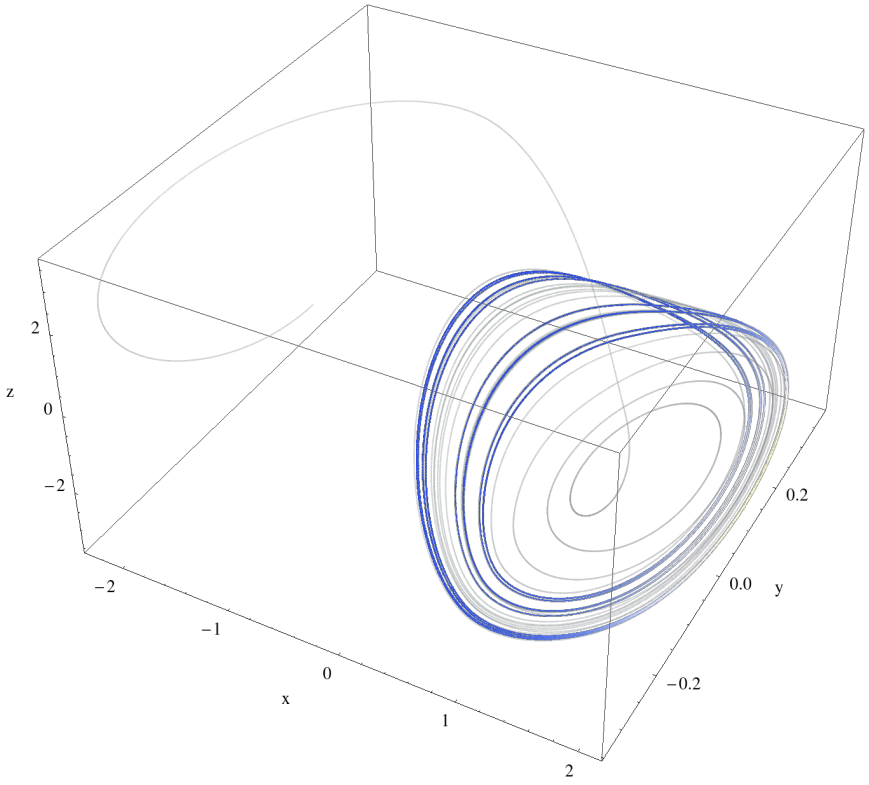


FIGURE 3.29: Figure of period 8 limit cycle using a self limiter with value 0.5.

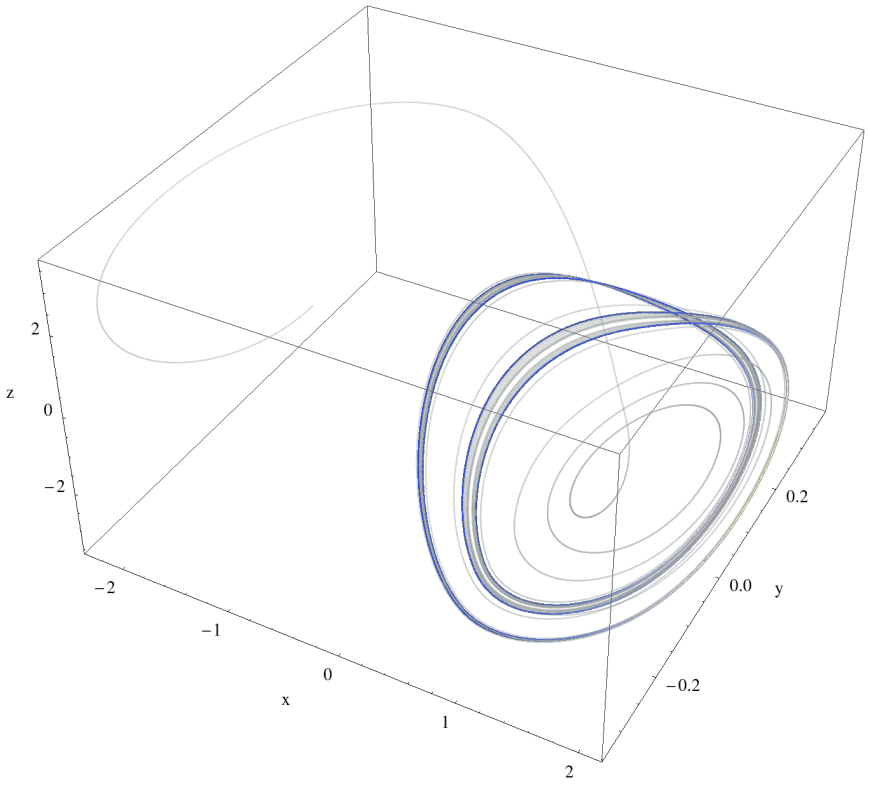


FIGURE 3.30: Figure of period 4 limit cycle using a self limiter with value 0.5.

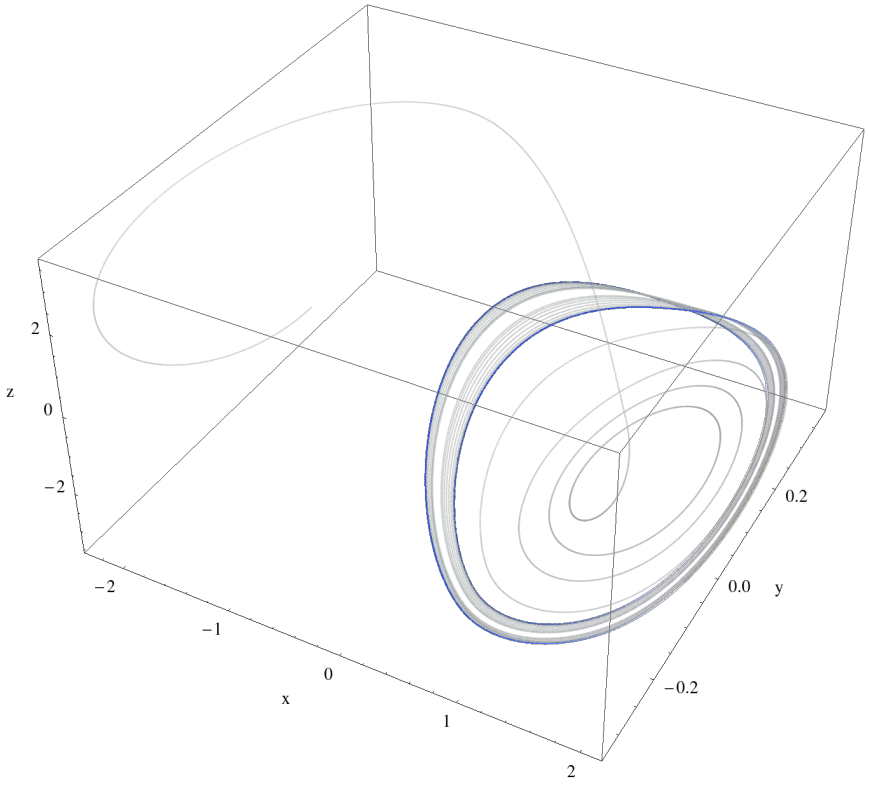


FIGURE 3.31: Figure of period 2 limit cycle using a self limiter with value 0.5.

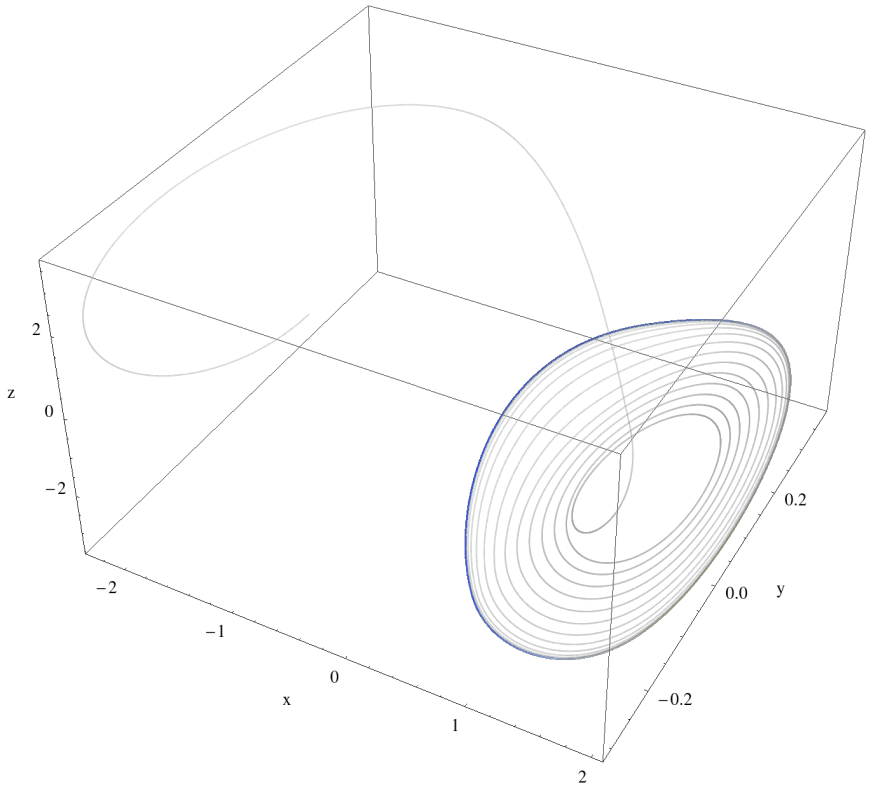


FIGURE 3.32: Figure of period 1 limit cycle using a self limiter with value 0.5.

The limit parameter can again be set beyond 0. The same plane rotation and spiralling behaviour can be seen, until it stays in the trajectory stays in the uppermost scroll of the multiscroll attractor before it spirals towards the limiter. There the trajectory can again be stabilized in the uppermost scroll into different periodic orbits (see table 3.5).

| *limitValue* | Limit Cycle |
|---|---|
| -0.109 | Period 16 |
| -0.115 | Period 8 |
| -0.15 | Period 4 |
| -0.2 | Period 2 |
| -0.37 | Period 1 |

TABLE 3.5: Different limit values resulting in trajectories of different periodicity in the uppermost scroll.

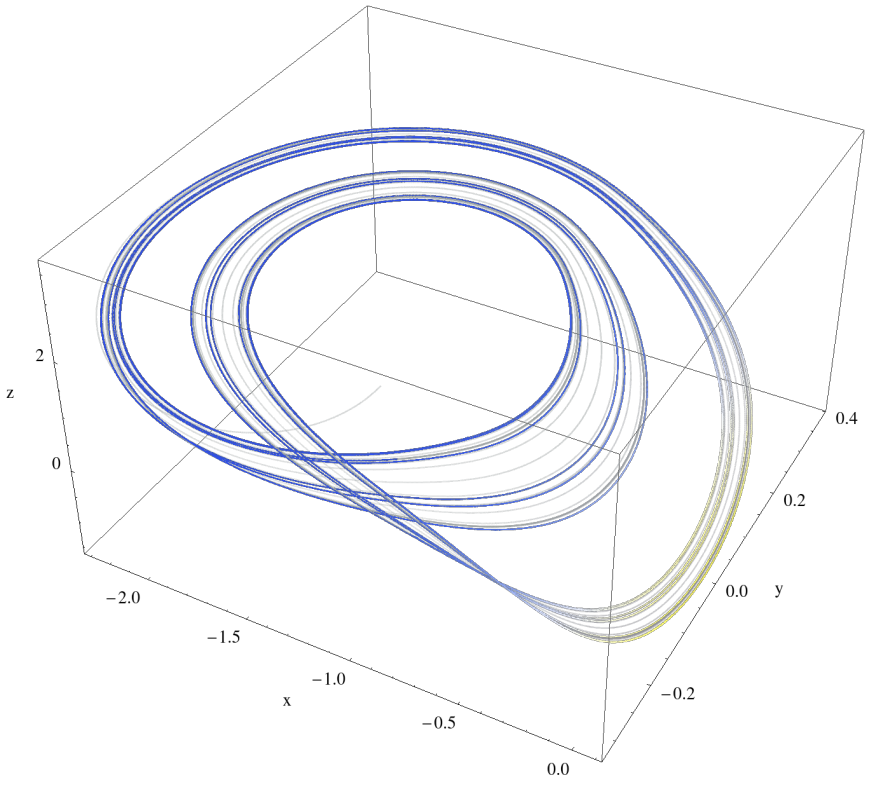


FIGURE 3.33: Figure of period 16 limit cycle in the upper scroll using a self limiter with value 0.5.

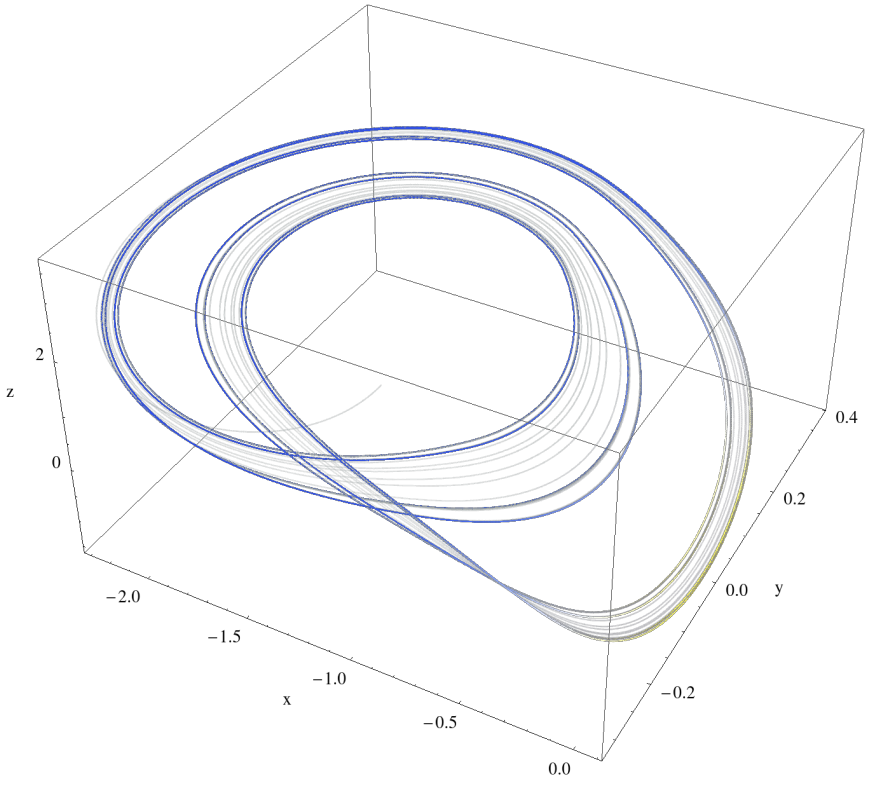


FIGURE 3.34: Figure of period 8 limit cycle in the upper scroll using a self limiter with value 0.5.

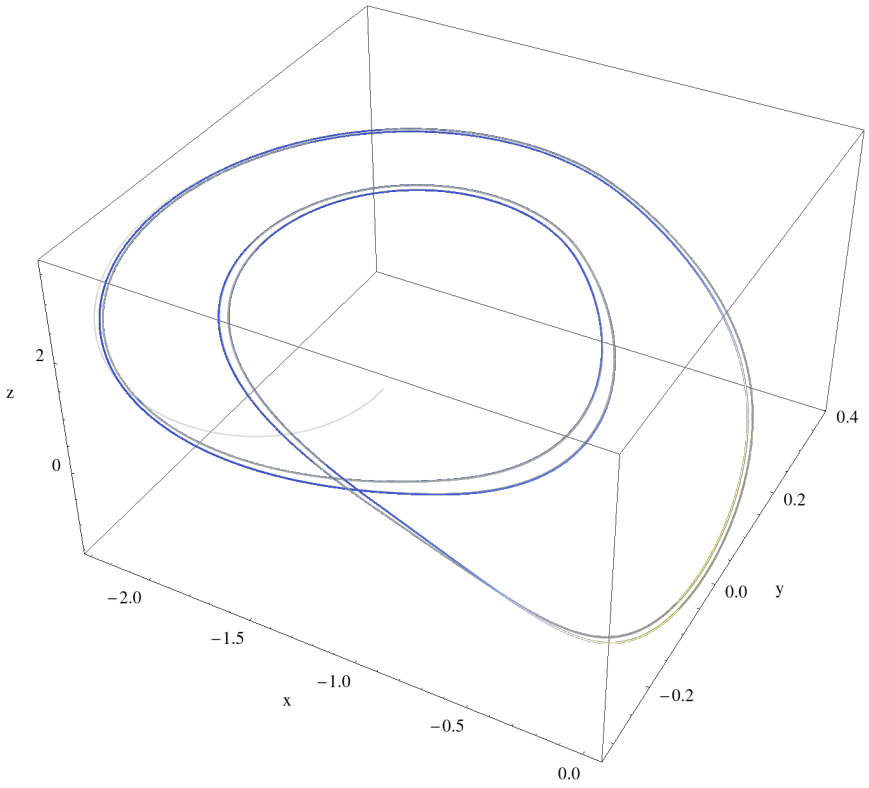


FIGURE 3.35: Figure of period 4 limit cycle in the upper scroll using a self limiter with value 0.5.

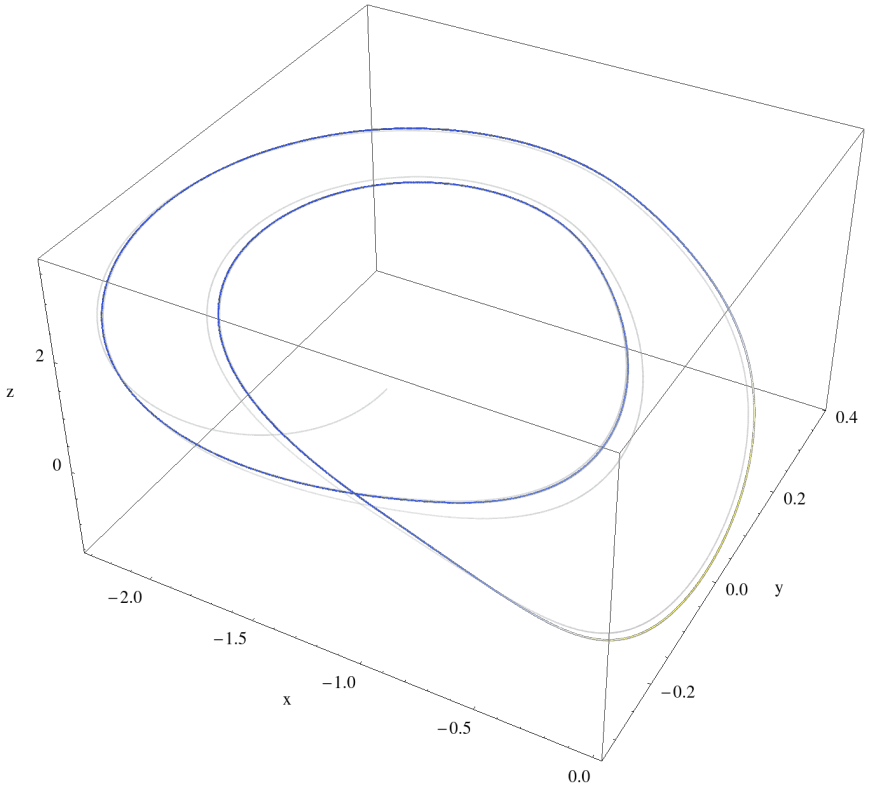


FIGURE 3.36: Figure of period 2 limit cycle in the upper scroll using a self limiter with value 0.5.

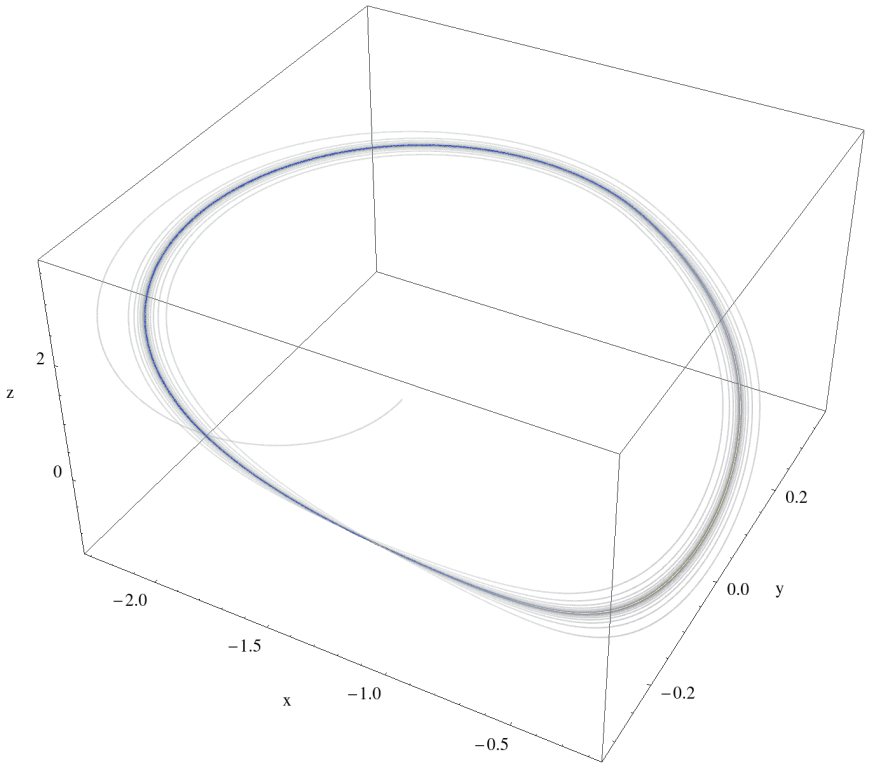


FIGURE 3.37: Figure of period 1 limit cycle in the upper scroll using a self limiter with value 0.5.

Pushing the limiter further into the negative limit values, one can again observe the plane rotation and the spiralling towards the uppermost scroll. Further limiting can suppress the progression of the trajectory entirely. The trajectory tries to spiral back into the uppermost scroll within limiter window $[-1, -4[$, but the radius of the spiral gets so small that the maximum precision of Mathematica is reached with limiter values $< -4$. Therefore the resulting trajectory starts to be discretized by the limited precision.

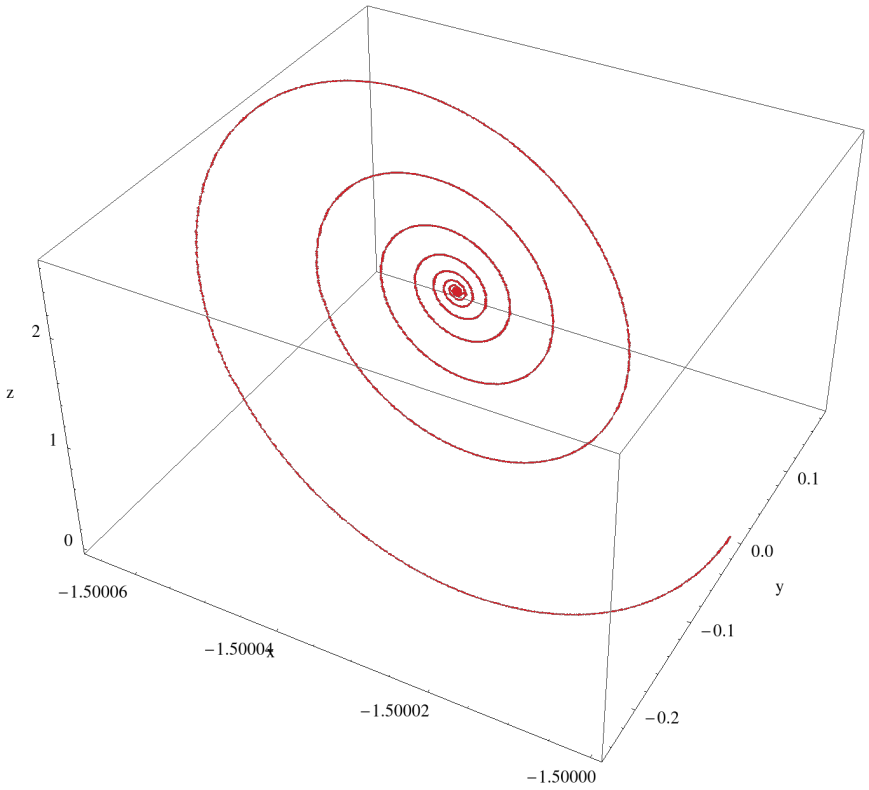


FIGURE 3.38: Figure of discretization artifacts in the uppermost scroll using a self limiter with value 0.5.

### 3.3.3 Chaotic Chua Circuit Controller

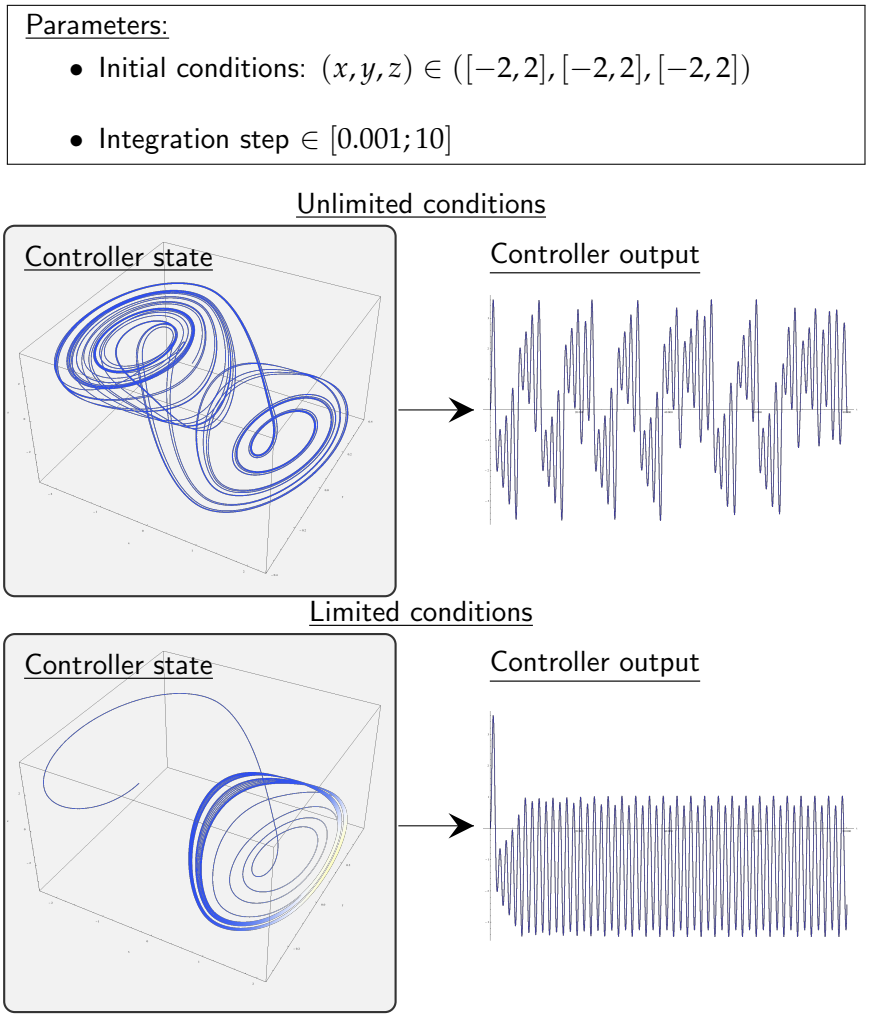


FIGURE 3.39: The specification of the chaotic chua controller, its internal state and output signal. The internal state of the chaotic chua controller is three dimensional because the Chua circuit's equations are defined using three dimensions. The output signal is chosen to be the z dimension of the internal state and is therefore only one dimensional. The controller state and output are shown for the unlimited condition and an example of limited condition. The example limitation leads to a period two limit cycle. If the chaotic chua circuit controller is mutated during the mutation step of the simulator, the parameters are chosen from a uniform distribution out of the respective intervals.

The chaotic controller used to control a single DoF in a joint of a creature is driven internally by the Chua circuit examined above. The Chua circuit's parameters are kept constant to keep the parameter space for the evolution small, also there exist many parameter conditions that do not lead to a chaotic ground state. Only the initial conditions are chosen randomly within the bounding box $(x, y, z) \in ([-2, 2], [-2, 2], [-2, 2])$ and the integration step is chosen randomly so that *integration step* $\in [0.001, 10]$. During a mutation of the chaotic chua circuit controller, the initial conditions and the integration step are chosen anew from a uniform distribution out of the respective intervals. Even though the integration step can vary

from controller to controller, the base integration step, is kept constant at *integration base* $= 0.0005$. The integration step then is performed by calculating small sub-steps of size of the *base integration step* until a smaller, residual integration step is left to calculate. Thereby, variable size integration speeds can be performed, but the actual base integration step is kept constant. The numerical integration again is performed using a Runge-Kutta order 6 integration scheme run in real-time. However, in the creature's controller, the simple limiter is not directly applied as in the experiment above, but it will be tested if the controller output can be limited by the joint morphology, so that periodic motion emerges not in the controller, but in the joint. If this does not prove to be successful, a feedback to the chaotic system will be implemented which limits it according to the current state of the controlled joint.

# Chapter 4

# Results

## 4.1 Evolved Creatures with Uncoupled Sinusoidal Controllers

In the first experimental setting, the sinusoidal controller as described in 3.2 was used to evolve creatures. A population of 100 creatures was set up with an average velocity fitness function and a minimal height fitness function. The latter was chosen because the simulator had issues in the physical model setup that led to creatures with the ability to fly due to imaginary forces (as described in 2.1.3). The minimal height fitness function punished those creatures that reached a high average height, which is the case for creatures that are flying. The simulation was run for approximately a day and took 181 generations for locomotion behavior to occur. Every creature was evaluated for 20 seconds. 181 generations of 100 creatures evaluated for 20 seconds take $\approx 4.2\,\mathrm{days}$, however, since the simulator can evaluate faster than real-time if the simulation's load permits it, it could simulate the creatures $\approx 4.2$ faster than real-time. The successful solutions evolved can be seen below in figures 4.1, 4.2, 4.3 and 4.4.

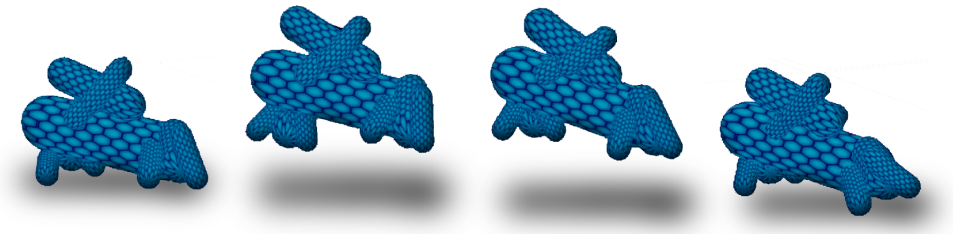

FIGURE 4.1: Figure of a jumper using sinusoidal controllers. Its four legs under the main body let it leap forward. A video of the jumping creature can be found on YouTube: `https://youtu.be/jMHQnsCy8_k`.

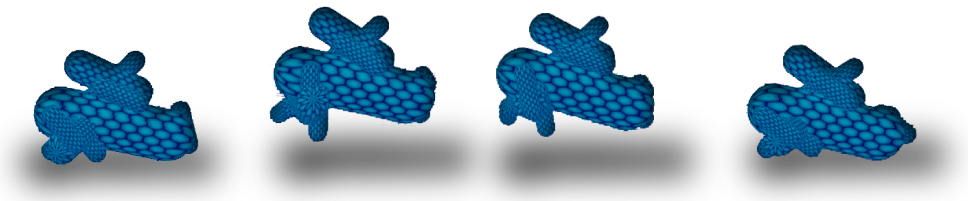

FIGURE 4.2: Figure of another jumper using sinusoidal controllers. Its two legs in the back propulse the body forward while the front contact point acts as a stabilization. A video of the jumping creature can be found on YouTube: `https://youtu.be/xc9uWrDYRic`.

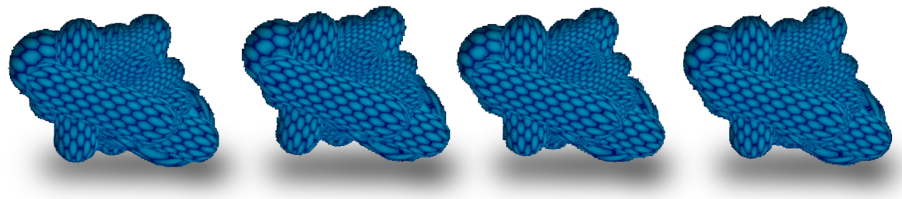

FIGURE 4.3: Figure of a walker using sinusoidal controllers. It has two legs in the back, which push the body forward. The front contact point again stabilizes the movement. A video of the walking creature can be found on YouTube: `https://youtu.be/PYMajm8A9eQ`.

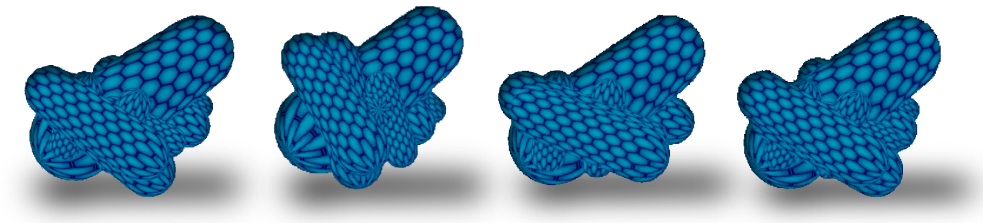

FIGURE 4.4: Figure of another walker using sinusoidal controllers. This creature has many legs on both sides, which synchronously touch the ground and make the body crawl forward. A video of the walking creature can be found on YouTube: `https://youtu.be/0AmdOUpvuPY`.

Creatures that exhibit successful locomotion seem to share a common ancestor, which might have been the first creature to evolve a successful walking pattern, since all of its offspring share a similar skin coloring. Since the skin color is randomly set for one body part, the common ancestor could only have had all the same color and must have been replicated using crossover. In crossover (as described in 2.1.6.2), subsections of the two participating creature genotypes are taken and combined into one new genotype. This process involves no mutations, therefore the crossover must have happened mainly among creatures sharing the same skin coloring. The elitism mechanism of the evolutionary process, passing the most successful creatures directly to the next generation, kept the first successful creature, that then generated an onset of an avalanche of successful creatures, flooding the population with very similar solutions.

The major drawback of the sinusoidal oscillator controller is that they do not show any adaption to the environment during the evaluation. The only way for the controller to adapt its control signal is on the evolutionary time-scale through mutation, which does not result in an advantage for the individual in its current evaluation, but only helps a future generation of the individual to eventually converge to a new, applicable solution.

## 4.2 Simple Limiter Control in the Simulator

The chaotic controller, introduced in the previous chapter, which uses an underlying chaotic system, is a marvellous source of periodic patterns of motion due to its infinite number of unstable periodic orbits. If a creature could use simple limiter control to stabilize UPOs and exploit the periodic orbits for locomotion patterns, this could lead to a more robust and adaptive way of motion. Experimentally, this would mean that if the creatures exhibited

periodic locomotion patterns, it could be shown that the successful creatures controlled the amount of chaotic movement of the controller or the chaotic movement of the body part and stabilized unstable periodic orbits to exploit it for body part motion. Showing that the controller or body part motion is less chaotic can be shown by calculating the largest Lyapunov exponent, which accounts for the amount of divergence of two slightly different initial conditions of the controller or the body part motion.

The experiments were run on the chaotic controller as described in 3.3.2. Simple limiters can come in various ways, since they only must be natural to the system at hand [2]. Therefore the first experiments were run using chaotic controllers without direct limiters implemented into the controllers. The second experiments then used sensory feedback to impose limiters on the chaotic controller.

### 4.2.1 Indirect Limiter Control through the Morphology

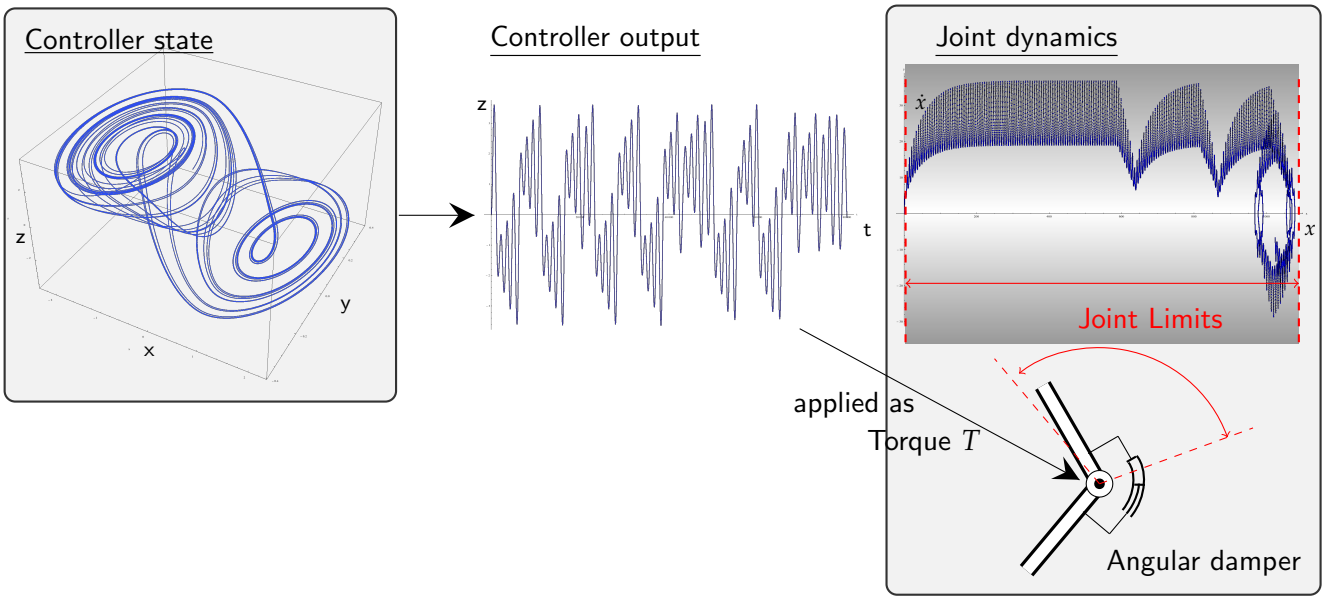


FIGURE 4.5: The indirectly limited chaotic controller and its interaction with the joint. The gradient in the joint dynamics plot represents the damping of the joint, where darker coloring means a stronger damping force. The limiters defined through the environment and the joint properties constrain only the movement of the joint, but do not influence the chaotic controller behavior itself. Therefore the environmental influences and the joint properties can not change the dynamics of the controller.

The first experiments use chaotic controllers without direct limiters. The controllers start with different initial conditions and different integration speeds, but are not limited using sensory input. Therefore the chaotic trajectories emitted from the controllers are only limited through the morphology.

In the following plot (Figure 4.6), the behavior of the model leg with an unlimited controller can be observed. The model leg continuously rotates into one direction until it hits the joint limit, from which it then moves back and forth in the case the controller signal's sign changes for a short time. However no periodic movement can be observed over the long run.

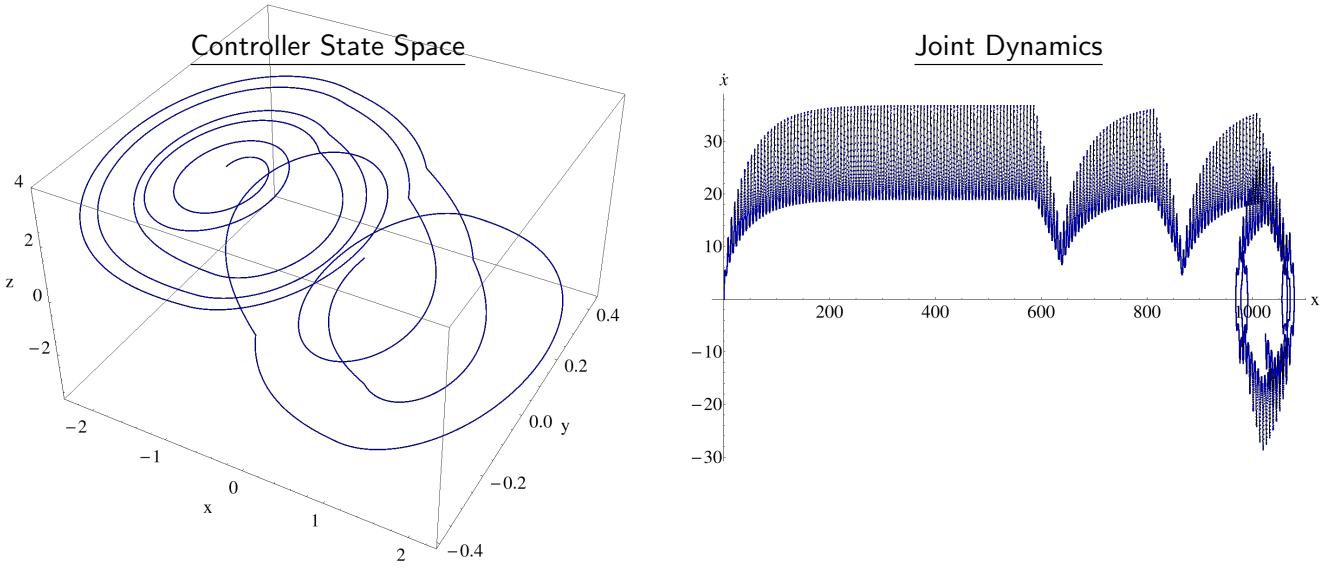


FIGURE 4.6: Unlimited chaotic controller controlling model leg. The velocity on the y axis fluctuates in the positive range and therefore, the position increases positively. At the end, the velocity switches from positive to negative, making the position oscillate as well. However, as the chaotic trajectory again switches to another limit cycle, the position increases again only into one direction.

| | |
|---|---|
| Gravity: 0g | Sensors: - |
| Output: z(t) $\rightarrow$ Joint torque | - |
| Torque scaling curve: 0.7 $(mass_1 \cdot mass_2)$ | - |

Since the model leg might be a bad predictor of how higher complexity could influence the resulting creature, 4 different evolutionary settings in terms of evaluation time and number of creatures in the population were run. However, the desired limiter control exhibiting periodic or quasi-periodic motion could not be observed in any of the creatures. This could be explained due to the lacking feedback from the body part to the chaotic system itself. That is why the chaotic system runs completely uninfluenced and is independent of the body part being limited in its movement. Therefore it was necessary to implement sensory feedback of the respective, controlled joint so that the limits of the morphology would impose limiters on the chaotic controller.

### 4.2.2 Direct Limiter Control through Sensory Feedback

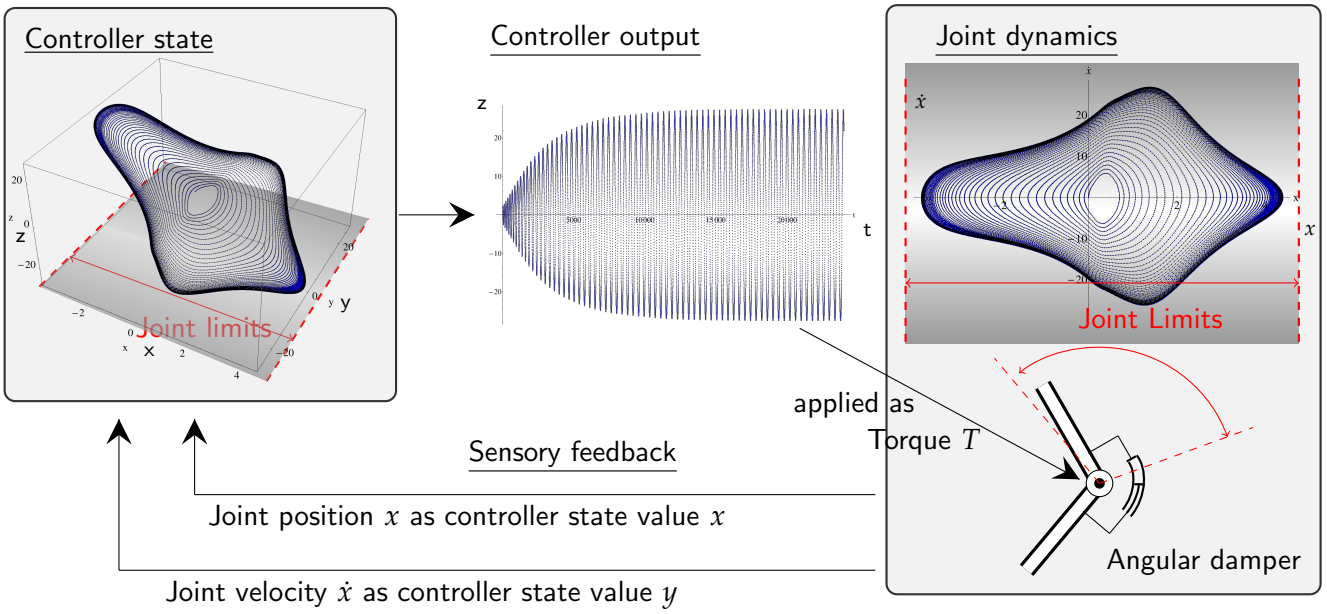


FIGURE 4.7: The directly limited chaotic controller and its interaction with the joint. The gradient in the joint dynamics plot represents the damping of the joint, where darker coloring means a stronger damping force. The limiters defined through the environment and the joint properties constrain both the movement of the joint and the phase space of the chaotic controller. Through the constraint within the phase space, the controller can not exhibit its full dynamics any more and it stabilizes in certain periodic oscillations defined by the parameters.

For the second experiment, a locally influenced chaotic controller was used. The chaotic controller output is ruled by three differential equations defined for $\frac{dx}{dt}$, $\frac{dy}{dt}$ and $\frac{dz}{dt}$ which influence the current state of the system stored in $x(t), y(t)$ and $z(t)$. In this experiment, the controller is limited using the position sensor and the velocity sensor of the joint degree of freedom the controller controls. As outlined below, the following limiter configuration was found experimentally: Before every integration step, the controller state $x(t)$ is replaced by the position sensor output and the controller state $y(t)$ is replaced by the velocity sensor output. The limiter system imposes limiters on two of the three dimensions in the situation where the body part motion itself is limited due to a collision with ground or other body parts. In case of a collision, the joint velocity of the DoF drops and the position comes to a halt. This limits the chaotic controller in two different ways, and it is in question what trajectories will be generated by the system. The limiter setup used in a model organism is then used to observe the influence of gravity, friction and damping on the chaotic system. If it is found to be experimentally successful and the trajectories exhibit periodic or quasi-periodic movement or interleaving periodic-chaotic sequences different from the usual intrinsic chaotic motion

of the underlying chaotic system, it will be used in evolutionary settings to evolve creatures controlled with it.

#### 4.2.2.1 Limiting the Model Leg in the Simulator

First, the different limiter configurations with z(t) as a torque output were tested on the model leg. The model leg is positioned slightly above the ground, so that if starting with zero gravity, the body part floats and moves freely without touching the ground. In the tables below, the friction values are therefore omitted when there is zero gravity. The initial conditions for the chaotic controller controlling are set to $(-1.5, 0, 2)$ in all experiments unless stated otherwise. When stated that the sensor X limits the x dimension, this means that the value x(t) of the dimension x is replaced with the X sensor value in every cycle. The following plots always show the phase space of the chaotic controller on the left side and the sensor space of the controlled joint DoF on the right side.

In the next plot, the dimension x is limited with the position sensor, while leaving the dimension y without limitation.

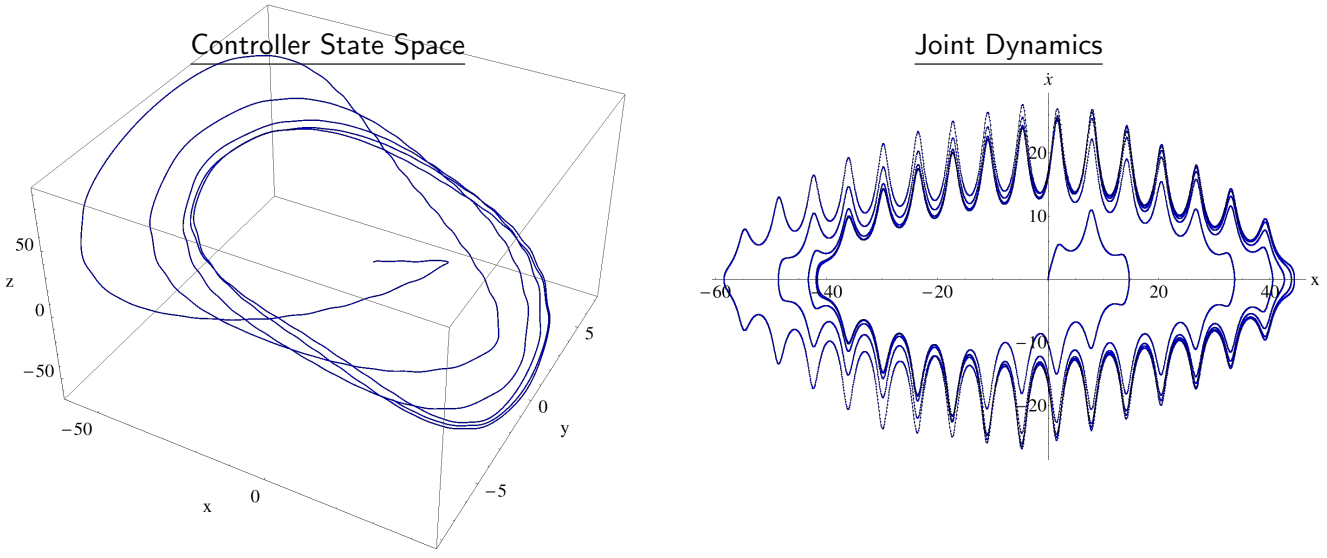


FIGURE 4.8: A strange quasi-periodic limit cycle can be observed in the chaotic controller phase space and the sensor space. The model body part oscillates strangely.

| | |
|---|---|
| Gravity: 0g | Sensors: Joint position → x(t) |
| Output: z(t) → Joint torque | - |
| Torque scaling curve: $0.7\ (mass_1 \cdot mass_2)$ | - |

When switching the limited dimension from x to y, the following strange trajectory can be observed in the sensor space.

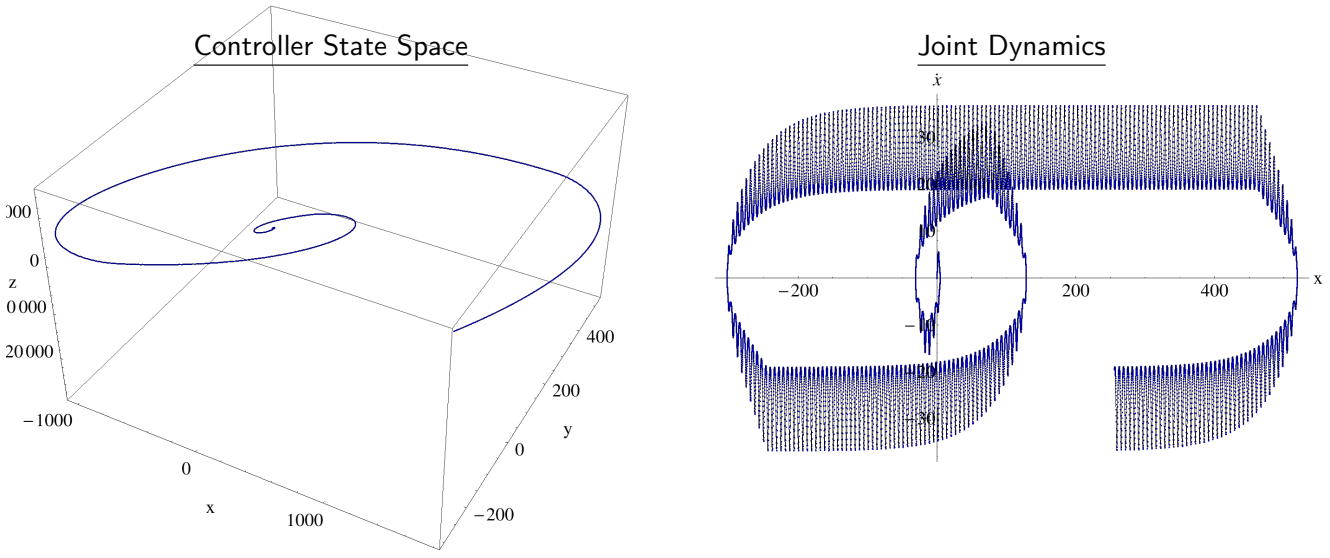


FIGURE 4.9: In the chaotic controller, one can observe an outward spiral, which indicates that the joint position limiting the y dimension turns the original attractor into a repeller.

| | | |
|---|---|---|
| Gravity: 0g | Sensors: | Joint position → y(t) |
| Output: z(t) → Joint torque | | - |
| Torque scaling curve: 0.7 $(mass_1 \cdot mass_2)$ | | - |

If the same dimensions are limited by the velocity sensor instead of the position sensor, a behavior as shown in figure 4.10 can be observed.

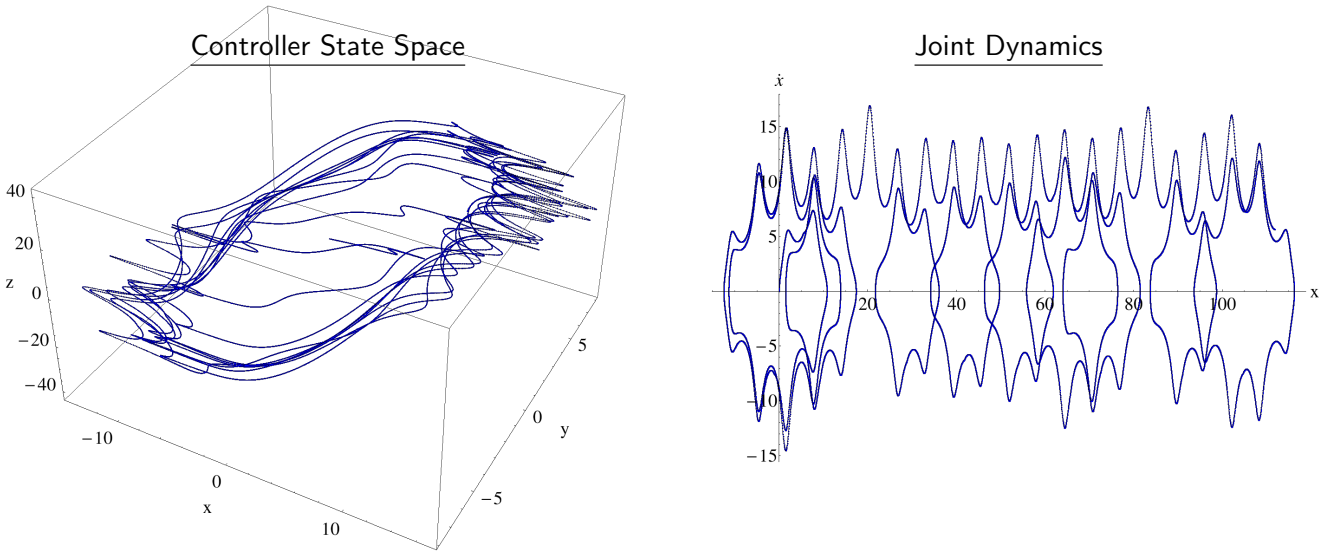


FIGURE 4.10: A strange quasi-periodic limit cycle can be observed in the controller phase space, but not in the sensor space. The velocity of the joint moves in a strange oscillation around zero, resulting in a net increase of the position.

| | |
|---|---|
| Gravity: 0g | Sensors: Joint Velocity $\rightarrow$ x(t) |
| Output: z(t) $\rightarrow$ Joint torque | - |
| Torque scaling curve: $0.7\ (mass_1 \cdot mass_2)$ | - |

If the joint velocity limits y(t) instead of x(t), the trajectory shows convergence to a limit cycle.

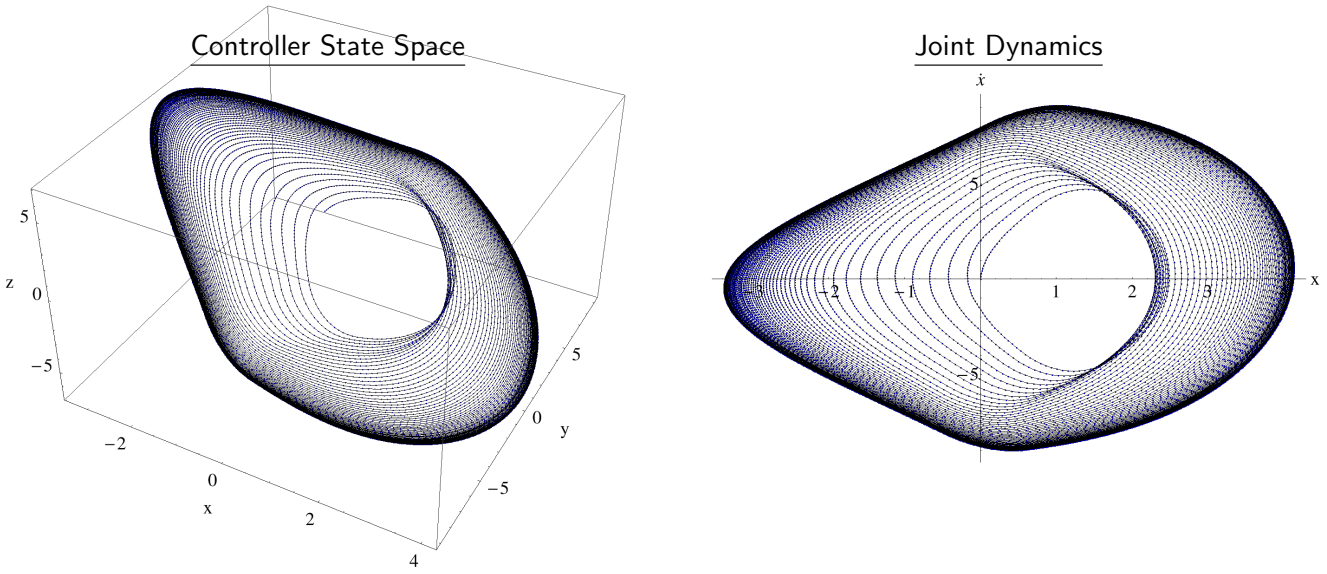


FIGURE 4.11: The trajectory performs symmetrical outward spiral which approaches a limit cycle of periodicity 1 at the outer bounds. This behavior can be explained by the fact that in the Chua equations, a simple negative feedback cycle is constructed. If the velocity increases into the positive direction, z(t) gets driven into the negative direction, therefore a torque pointing into the opposite direction is applied. Additionally a descent down into the plane of the periodic orbit can be observed.

| | |
|---|---|
| Gravity: 0g | Sensors: Joint Velocity $\rightarrow$ y(t) |
| Output: z(t) $\rightarrow$ Joint torque | - |
| Torque scaling curve: $0.7\ (mass_1 \cdot mass_2)$ | - |

If two successful limiters get combined, namely the Joint position $\rightarrow$ x(t) resulting in a strange limit cycle and Joint Velocity $\rightarrow$ y(t) resulting in a smoother limit cycle, the resulting limit cycle looks as follows.

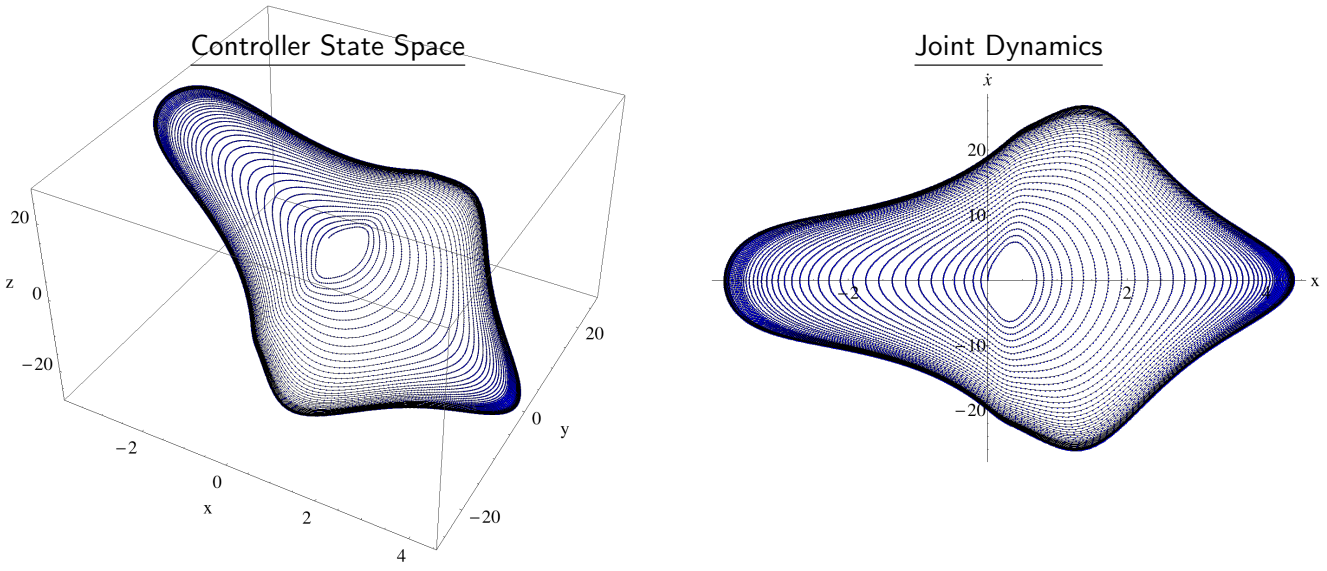


FIGURE 4.12: The shape of the limit cycle gets distorted.

| | | |
|---|---|---|
| Gravity: 0g | Sensors: | Joint position → x(t), |
| Output: z(t) → Joint torque | | Joint Velocity → y(t) |
| Torque scaling curve: 0.7 $(mass_1 \cdot mass_2)$ | | - |

However the combination of non-stabilizing limiters does not stabilize any unstable periodic orbit.

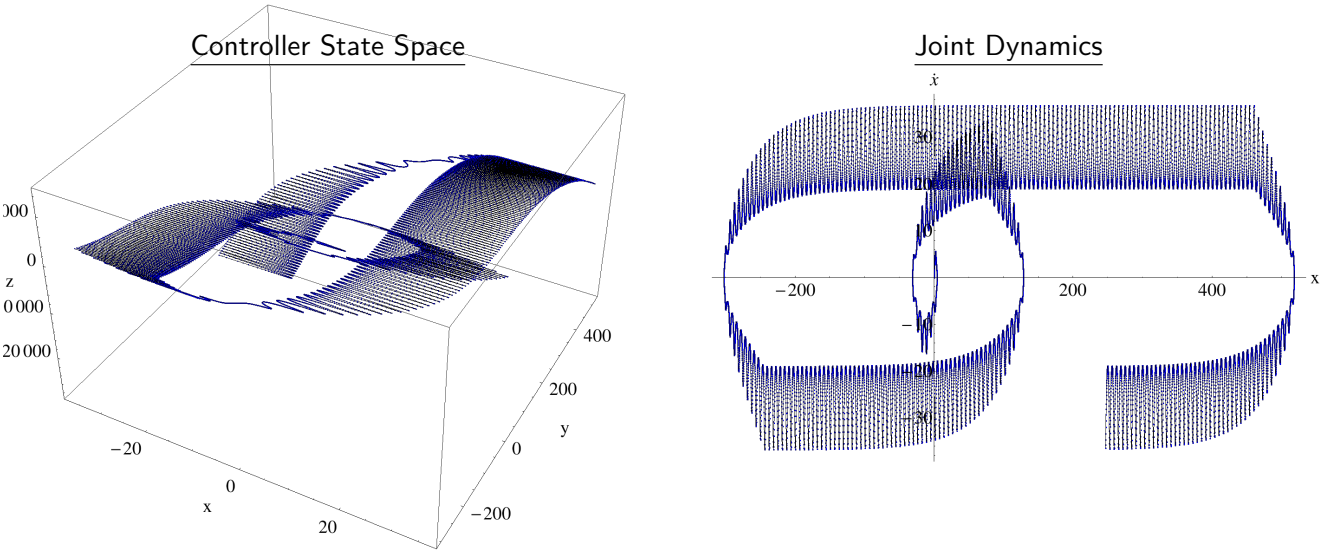


FIGURE 4.13: A strangely oscillating trajectory can be observed.

| | | |
|---|---|---|
| Gravity: 0g | Sensors: | Joint Velocity → x(t), |
| Output: z(t) → Joint torque | | Joint Position → y(t) |
| Torque scaling curve: 0.7 $(mass_1 \cdot mass_2)$ | | - |

The configuration where the x dimension is limited with the position sensor and the y dimension is limited with the velocity sensor is chosen for further examination. Based on this configuration, the model leg with different joint torques, friction and damping coefficients was simulated to observe the resulting change in controller and joint dynamics. In the following plots, different influencing factors on the limit cycle are shown:

**Interaction with Ground** The ground imposes a very hard limiter onto the creature. The movement of the creature gets instantly stopped, whereas the ground has infinite mass and therefore is not influenced by the creature. In some cases, the ground should therefore directly act as a very hard limiter onto the chaotic circuit, controlling the chaotic movement into low periods and orbits of smaller diameter as in the experiments performed in the section 3.3.2.1. In the first setting, the model leg is dropped onto a frictionless ground.

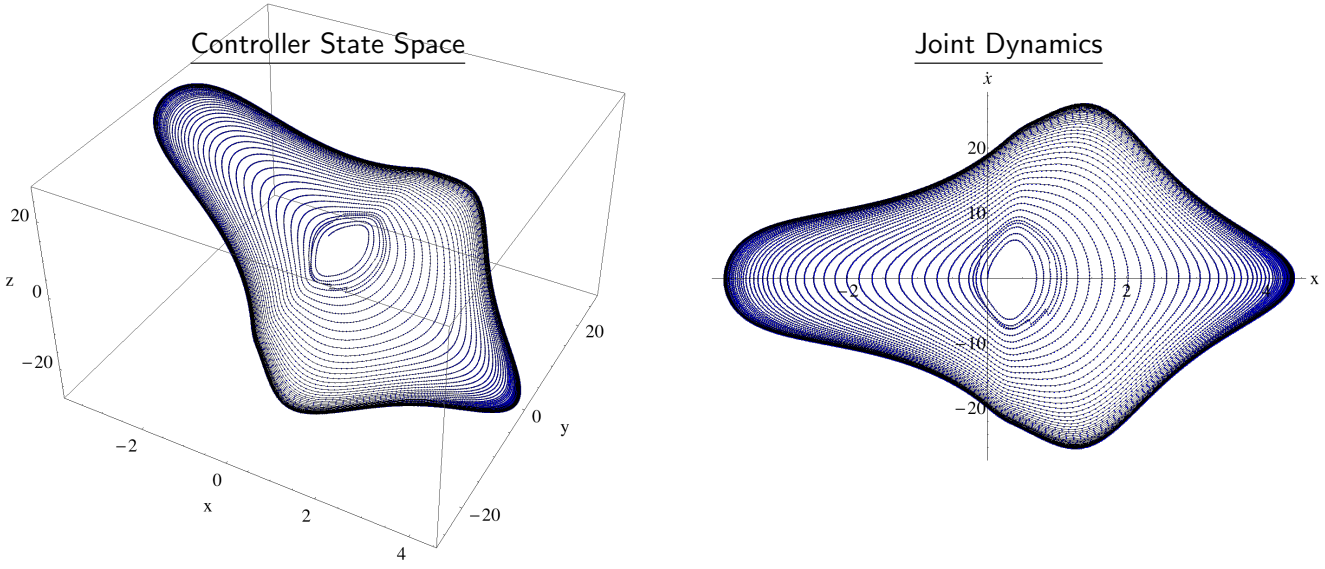


FIGURE 4.14: The interaction with ground can be observed both in the controller phase space and the sensor space. Small perturbations causing a drop of the velocity towards zero change the trajectory of the spiral. The frictionless ground limits the controller only quickly, because afterwards the model leg falls to the side, slides on the ground and converges to the original periodic orbit.

| | |
|---|---|
| Gravity: 1g | Sensors: Joint position $\rightarrow$ x(t), |
| Output: z(t) $\rightarrow$ Joint torque | Joint Velocity $\rightarrow$ y(t) |
| Torque scaling curve: $0.7\ (mass_1 \cdot mass_2)$ | Friction: Creature 0, Ground 0 |

Changing the friction of the ground to a non-zero value increases the limiter imposed on the controller.

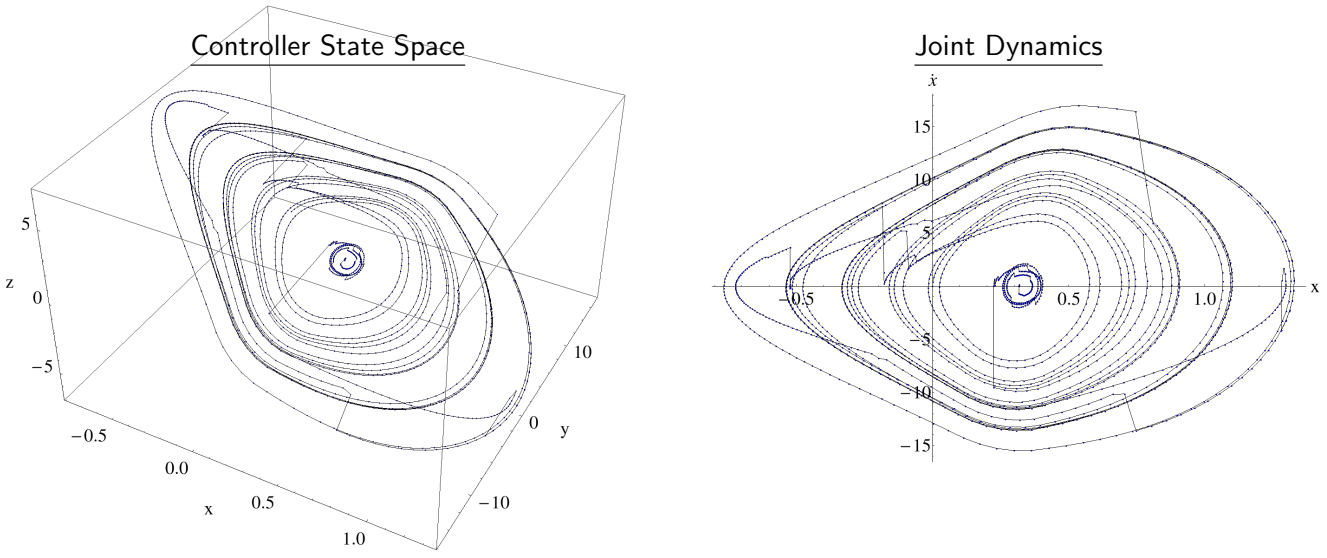


FIGURE 4.15: The ground with friction now shows a much more influential limiter. The model leg hits the ground several times while oscillating, which stabilizes several periodic orbits for one to two cycles. Finally the model leg comes to a rest as it blocks the oscillation as it falls to the ground while completely extended.

| | | |
|---|---|---|
| Gravity: 1g | Sensors: | Joint position → x(t), |
| Output: z(t) → Joint torque | | Joint Velocity → y(t) |
| Torque scaling curve: 2 $(mass_1 \cdot mass_2)$ | Friction: | Creature 1, Ground 1 |

The collisions with the ground are visible in both the controller state space and the joint sensor space as smaller or larger jumps. The collision causes first a drop in velocity and at the same time stops the displacement in position. In the plot 4.14, the collisions only cause a light displacement since the ground is frictionless, therefore the controller can converge onto the outer limit cycle. In the plot 4.15, where friction to the creature and ground is added, the controller can no longer converge to the outer limit cycle, but it stays on several smaller limit cycles until the morphology finally gets trapped and can not move any more. Both plots show how the controller can be robust or sensitive to limiters imposed by the environment.

**Influence of Damping** Damping is an indirect soft limiter applied to the physical joint system of the simple form: $F_{damping} = c \cdot \dot{x} = c \cdot v$. However, since the joint space is coupled to the chaotic controller space through the sensors, one can also observe the influence of the damper onto the controller.

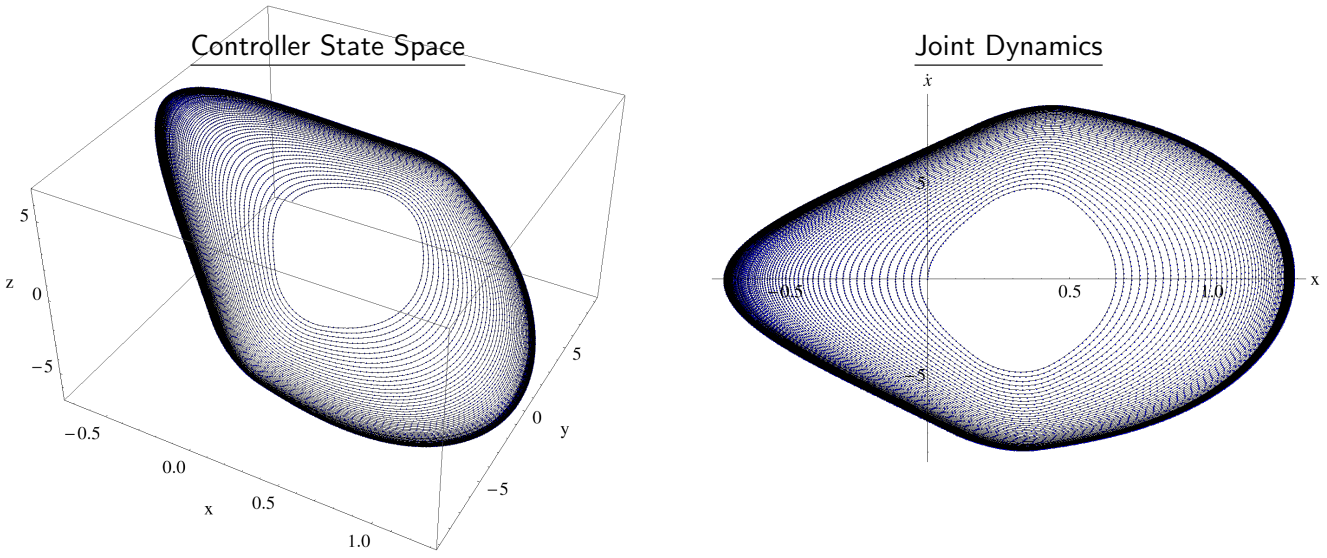


FIGURE 4.16: The damping of the model leg softens the curvature of the periodic orbit.

| | | |
|---|---|---|
| Gravity: 0g | Sensors: | Joint position → x(t), |
| Output: z(t) → Joint torque | | Joint Velocity → y(t) |
| Torque scaling curve: 0.7 $(mass_1 \cdot mass_2)$ | Damping: | 0.05 |

Gradually increasing the joint torque shows that the convergence to the limit cycle is increased and the original curvature of the periodic orbit comes back.

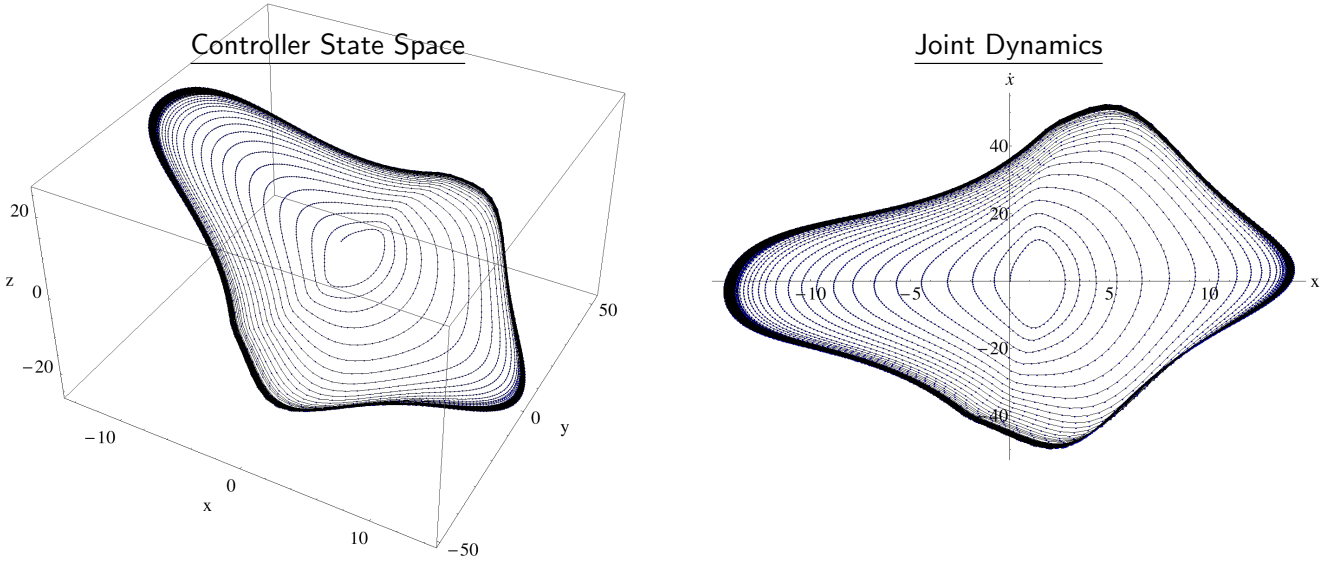


FIGURE 4.17: The convergence speed of the trajectory to the periodic orbit is increased, additionally the system again convergence to a curvature similar to the one without damping.

| | | |
|---|---|---|
| Gravity: 0g | Sensors: | Joint position → x(t), |
| Output: z(t) → Joint torque | | Joint Velocity → y(t) |
| Torque scaling curve: 4 $(mass_1 \cdot mass_2)$ | Damping: | 0.05 |

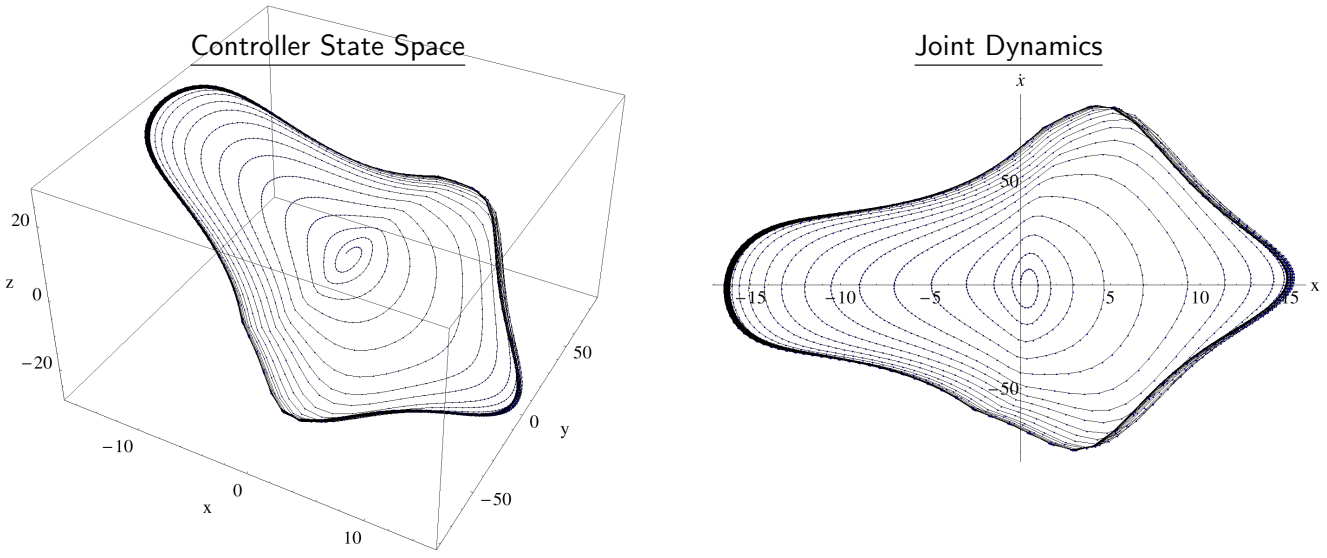


FIGURE 4.18: Increasing the torque again increases the convergence speed again.

| | | |
|---|---|---|
| Gravity: 0g | Sensors: | Joint position → x(t), |
| Output: z(t) → Joint torque | | Joint Velocity → y(t) |
| Torque scaling curve: 10 $(mass_1 \cdot mass_2)$ | Damping: | 0.05 |

If more torque and more damping is applied, a higher convergence speed can be observed. The trajectory again stays in a much smaller sub-manifold.

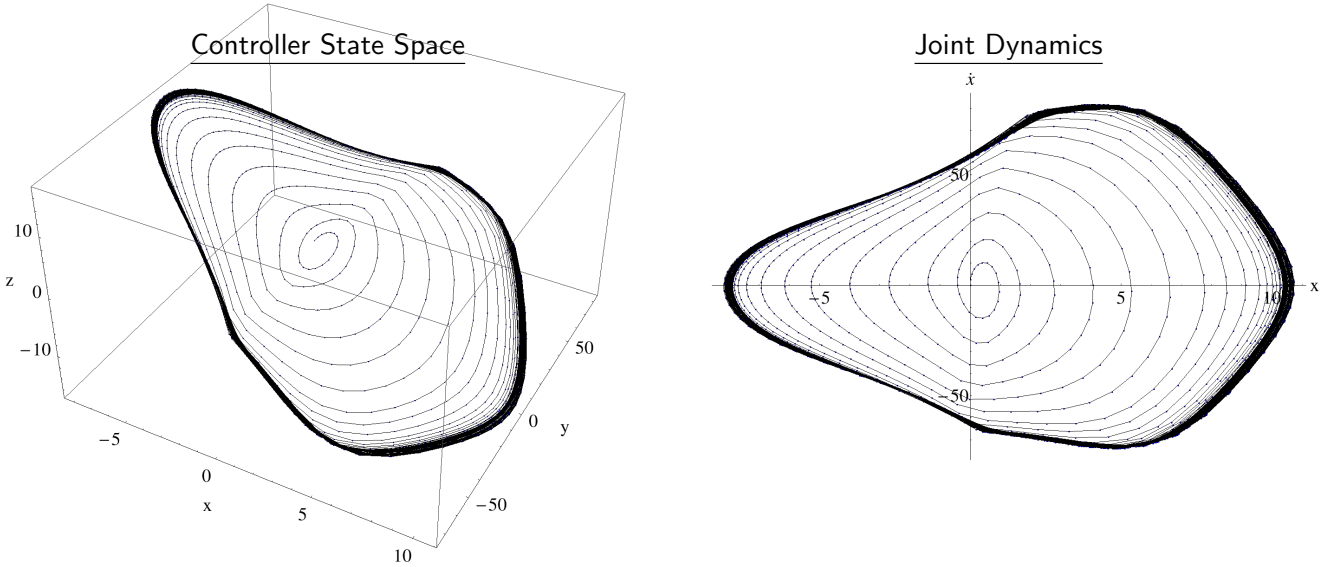


FIGURE 4.19: With more torque and more damping, the trajectory converges quickly to the periodic orbit.

| | | |
|---|---|---|
| Gravity: 0g | Sensors: | Joint position → x(t), |
| Output: z(t) → Joint torque | | Joint Velocity → y(t) |
| Torque scaling curve: 20 $(mass_1 \cdot mass_2)$ | Damping: | 5 |

Conclusively, the damping of the system leads to a smoother limit cycle of the original system's trajectory, however, still converges to a limit cycle, even if more force and more damping is applied.

**Influence of the X Dimension Limiter** From the initial experiments on limiter configuration (4.12), it is known that the velocity sensor limiting the y dimension could have been sufficient for the limiter control of the chaotic system. One can now observe the influence of the x dimension limiter when it is removed from the system. In this case, the initial conditions were set to $[0, 0, 0]$ to observe the complete trajectory.

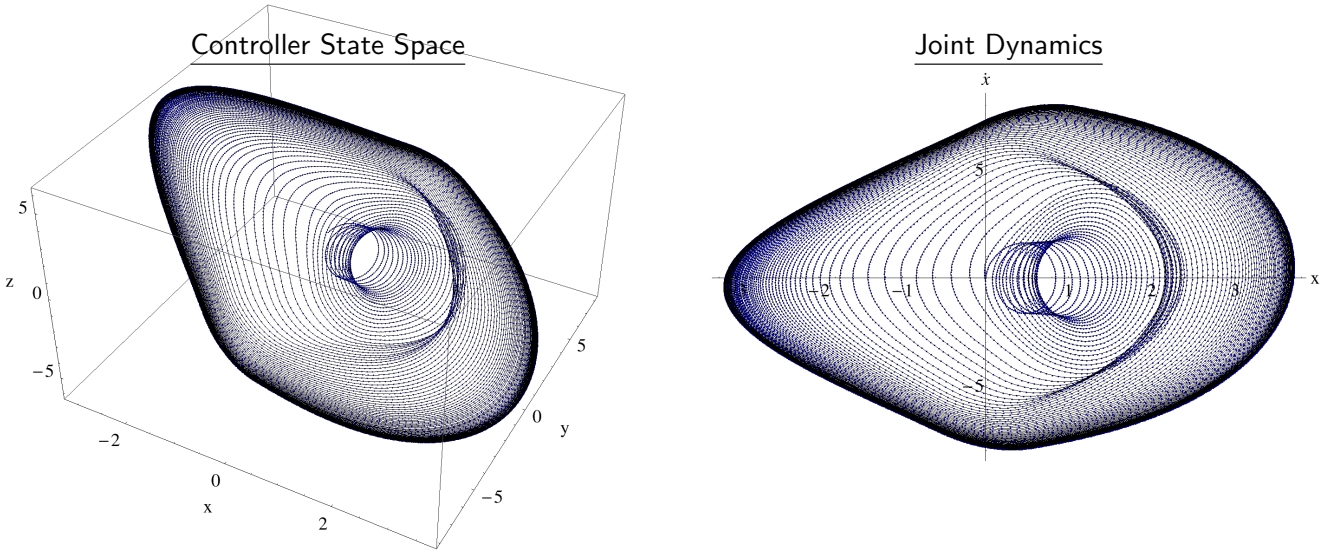


FIGURE 4.20

| | | |
|---|---|---|
| Gravity: 0g | Sensors: | Joint Velocity → y(t) |
| Output: z(t) → Joint torque | | - |
| Torque scaling curve: 0.7 $(mass_1 \cdot mass_2)$ | Damping: | 0.05 |

If the system gets one of its degrees of freedom back, it shows some of its internal dynamics. However, the original convergence behavior still occurs.

In conclusion it can be said that the stronger a chaotic system is limited through the output from another dynamical system, the more it exhibits similar patterns as the dynamical system it is limited by. Since in these experiments the Chua circuit is under a constant limiter from the joint dynamics, it can only emit a limited range of mostly periodic trajectories.

#### 4.2.2.2 Evolved Creatures with Direct Limiter Control

The setup to evolve creatures integrating the chaotic controllers was intended to be an exploratory one. In order to observe examples of creatures that locomote using chaotic controllers, three simulators with each a separate evolutionary setting were started. Each setting has 100 individuals in one population, each with the same average velocity fitness function. This setup was chosen through the single fitness function in a very liberal way, so that different creatures could evolve their own dynamics, because so many locomotion solutions are possible to be evolved. The simulators were run on a quad-core processor system, so the multiple simulators exploited the multiple cores much better than a single simulator could. The experiment using the direct limiter controller chaotic controller sounded much more promising, since the controller stayed in a quasi-periodic motion for all the time for almost all the model leg setups. Its output seemingly could be easily compared to the sinusoidal controllers. This assumption turns out to be quite false in practice, because when integrated into random morphologies by the evolutionary algorithm, the controller exhibits periodic motion at some point when limited appropriately. But in other cases, the controllers switches to a different periodic or non-periodic trajectory when influenced by the environment. Thus, a controller might sometimes stabilize a periodic orbit at one point, but remain in a chaotic state at all others.

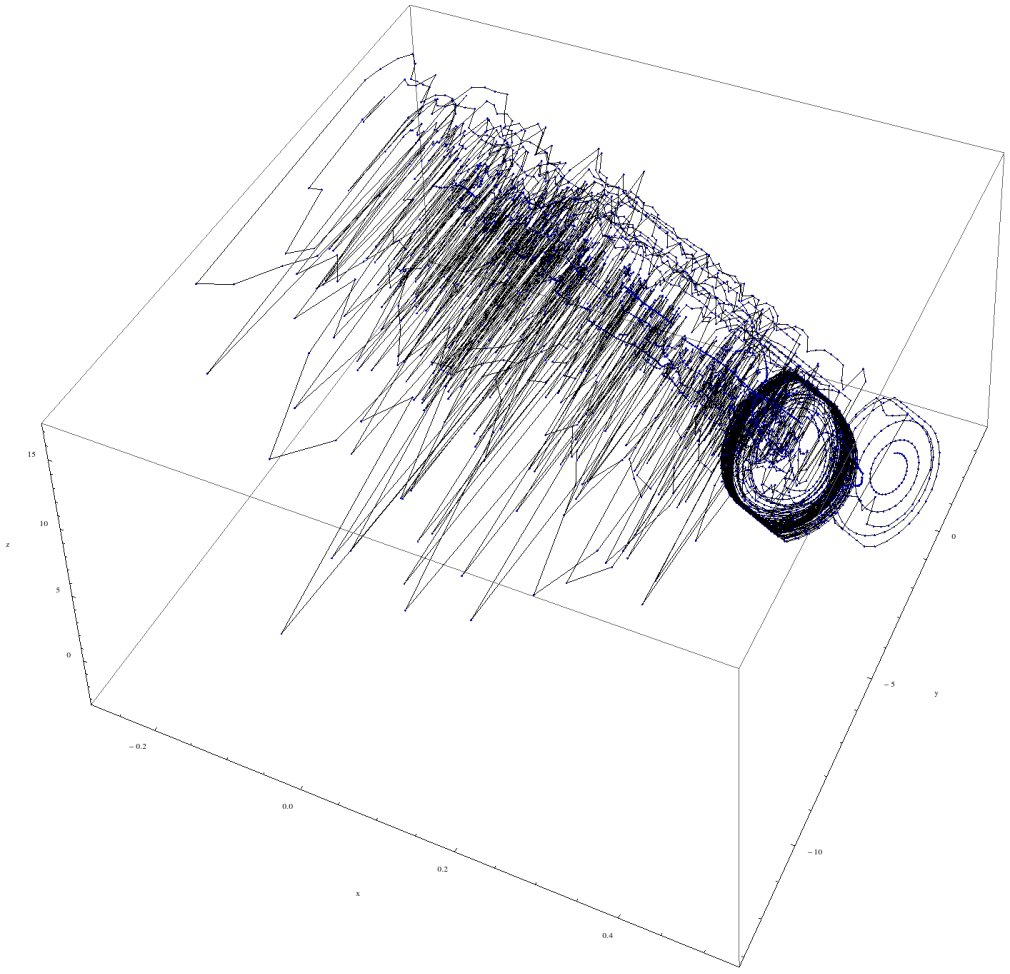


FIGURE 4.21: A plot of the controller phase space showing a trajectory containing interweaving chaotic and periodic phases. The phase transition of the controller happens when the limiters of the joint change and hence the chaotic system's phase space is increased or decreased. In the more limited case, the controller goes into low periodic oscillations, in the less limited case, it performs a chaotic trajectory.

The simple limiters of the environment apparently stabilize a certain periodic orbit, but change, when the environment changes. The controller therefore satisfies the idea of being adaptive to the environment, however it stayed in question if multiple controllers could be combined with a co-evolved morphology so that a satisfactory locomotion pattern would arise. The fitness function of average velocity totally leaves it up to the individual, how the movement would be achieved. Evolutionary algorithms are very exploitive, meaning that the evolved solutions easily exploit everything they can find to satisfy the fitness functions. However, the strength of the evolutionary algorithm, its versatility to find solutions, turns in this case into a very messy task for the experimenter. For higher order tasks such as locomotion, it means that one has to come up with a simple but precise description of the solution space using fitness functions. The definition must be general enough to accept a variety of solutions, but must be precise enough to accept only the desired solutions. What sounds easy for simple tasks, is not very easy for more complex tasks. One might argue

here that the question here is really what locomotion actually is. When defined as being an action causing directed movement, then nearly everything is allowed to be evolved. On the other end, restricting locomotion to body parts being moved in coordination to form complex oscillatory gaits, then only very few solutions fit the description. In this thesis, the definition was kept very loose, so that most solutions involving oscillatory actions causing directed movement can be considered locomotion. Therefore, the first simulations were started with the average velocity fitness function only. After 500 generations of simulated evolution, the simulators were stopped and the evolved creatures were examined. The definition was not at all sufficient to evolve locomoting creatures. Some of them were moving, but evolved into larger balls with many legs, so that the oscillatory movement caused the structure to roll. Some of them even just exploited the fact that growing tall and falling over causes them to slide a certain distance, resulting in a substantial average velocity. Because the results were not desirable, the experiment was redesigned to cope with the definition problem. The definition was changed multiple times after examining the evolved creatures. As long as the creatures showed hints to converge onto the desired behavior, the simulation was continued without changes, otherwise, the definition was changed again. The best results were achieved using a stronger restriction on the solutions, which was a combination of the average velocity fitness function and a minimal contact point quantity fitness function. The minimal contact point quantity fitness function reduced the the number of feet a creature develops, therefore a clearer locomotion pattern had to emerge. In the following figures, three of these evolved creatures are examined in detail.

**Walker** The first creature is a walker. It has two large feet, of which it always moves one forward while using the other as a stabilization.

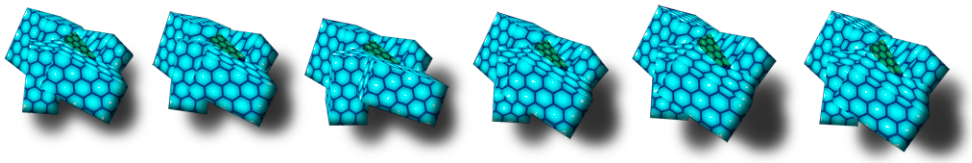

FIGURE 4.22: A figure of a walker featuring 4 chaotic controllers and 4 joints. It has two large feet it stands on, which push the body forward. In the front of the creature there is a stabilizing block that in certain cases touches the ground if the creature tips during its gait. A video of the walking creature can be found on YouTube: `https://youtu.be/jwaEXxB7ZB4`.

The walking creature features quasi-periodic orbits if the body parts are not limited by the ground. The quasi periodicity can be explained by the interdynamics generated by the different body parts moving simultaneously, therefore no single body part can follow a proper limit cycle. The joint limits cause the joint dynamics to be periodic.

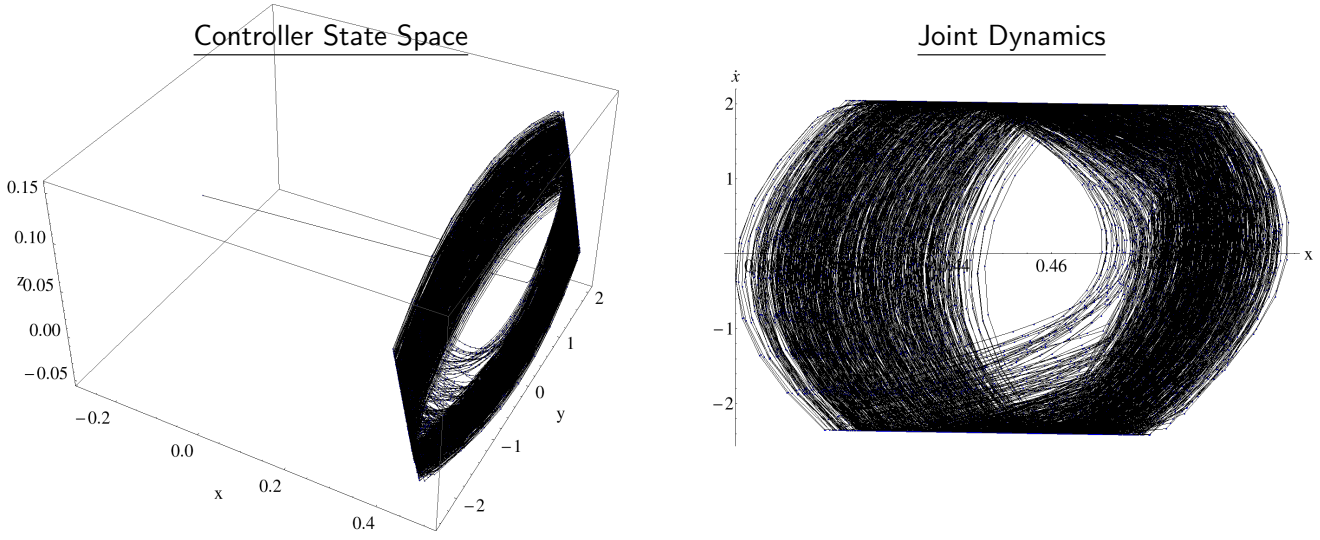


FIGURE 4.23: Dynamics of a chaotic controller and joint if the walker is off ground. The joint limits make the joint movement stay periodic even in the absence of the ground limiter.

If the same creature is put on ground, the same controller features chaotic and quasi-periodic dynamics.

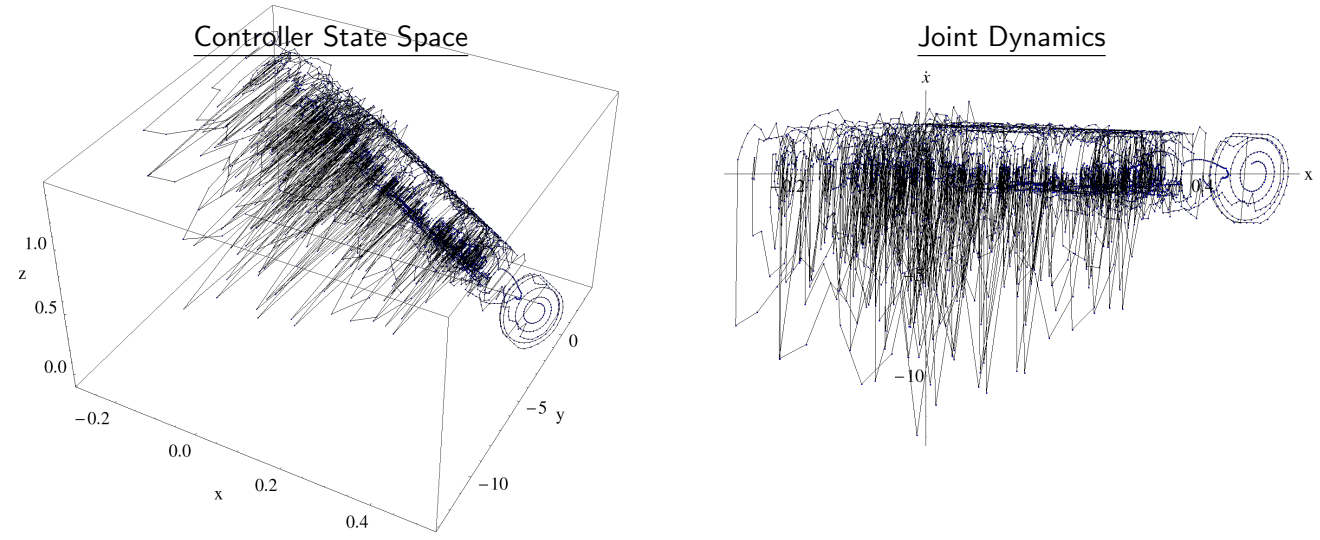


FIGURE 4.24: Dynamics of a chaotic controller and joint if the walker is on ground. The ground limiter makes the the velocity fluctuate strongly because the creature's limbs are shuffled on the ground. Therefore it is hard to observe a limit cycle.

The creature's behavior can be explained because the gait of the creature always traps one leg while moving the other, then the behavior switches to the other leg. The shuffling of the legs on the ground make the speed fluctuate strongly. This leads to a fluctuating trajectory

in the controller when the leg is completely trapped, and to a periodic trajectory when untrapped.

**Crawler** The second creature is a crawler. It moves on ground using a gait similar to a snake by oscillating its tail.

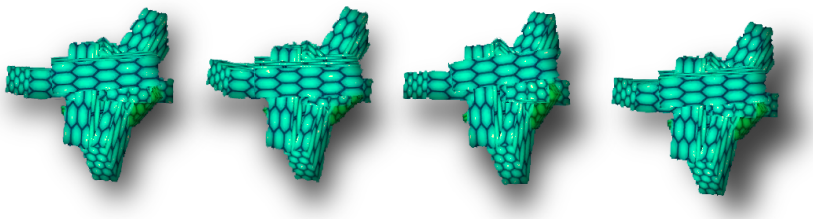

FIGURE 4.25: Top-down view of a crawler featuring 10 chaotic controllers and 10 joints. The crawler has a tail and two arms looking to the left and right from the head, and moves forward using a snake-like locomotion gait. A video of the crawling creature can be found on YouTube: `https://youtu.be/3qpJOooM4Vc`.

When off ground, the controllers of the creature show a regular oscillation behavior.

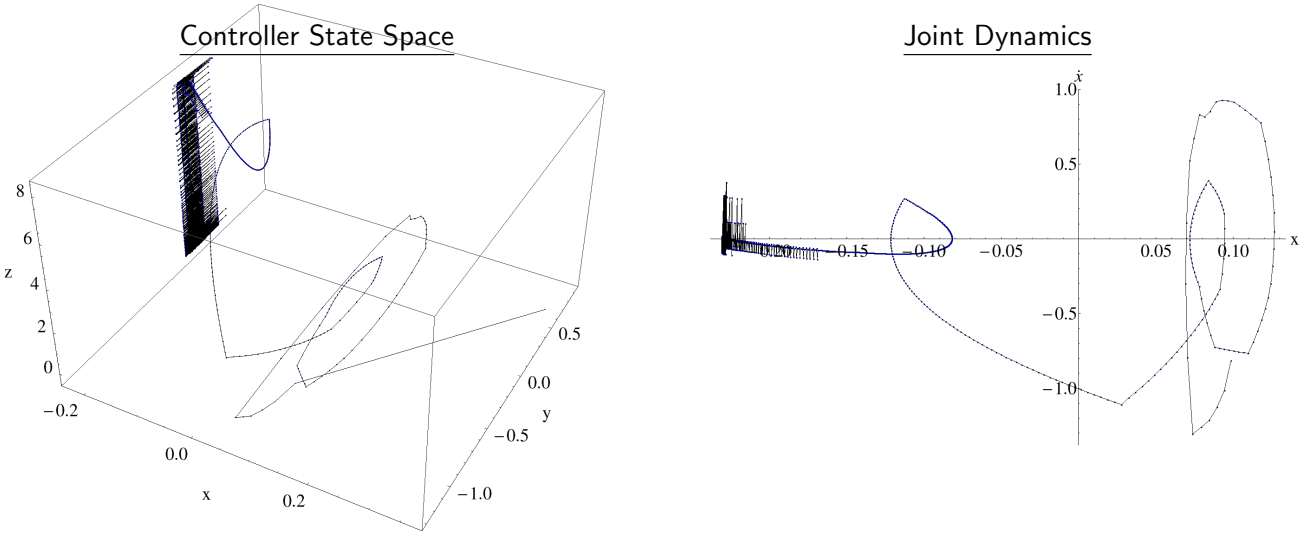


FIGURE 4.26: Dynamics of a chaotic controller and joint if the crawler is off ground. The oscillation behavior of the tail's controller is pretty regular.

The same controller features a much less regular behavior when on ground. Still the creature moves forward in a pretty regular manner. Therefore the plot might again be noisier because of the contact with the ground, which makes the joint velocity fluctuate.

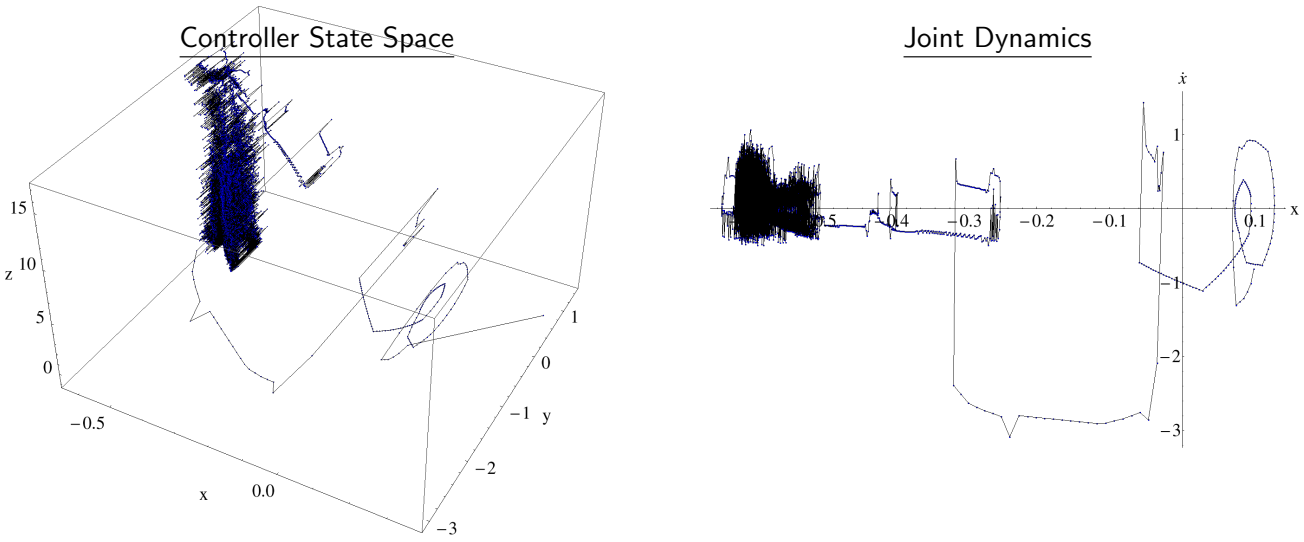


FIGURE 4.27: Dynamics of a chaotic controller and joint if the crawler is on ground. The joint velocity again fluctuates as with the other creature because it is in contact with the ground.

**Jumper** The third creature is a jumper. It moves on ground by moving its body parts to gain momentum, which makes the main body finally lift off and take a small leap forward.

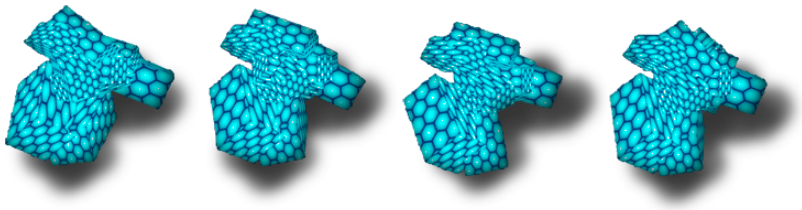

FIGURE 4.28: Figure of a jumper featuring 10 chaotic controllers and 10 joints. The jumper has several legs it uses to accumulate momentum in order to jump forward. A video of the jumping creature can be found on YouTube: `https://youtu.be/JyBFdD_wCoQ`.

When off ground, the controllers of the creature show a chaotic oscillation behavior only limited by the two joint limits at $-2$ and $1$ in the x dimension.

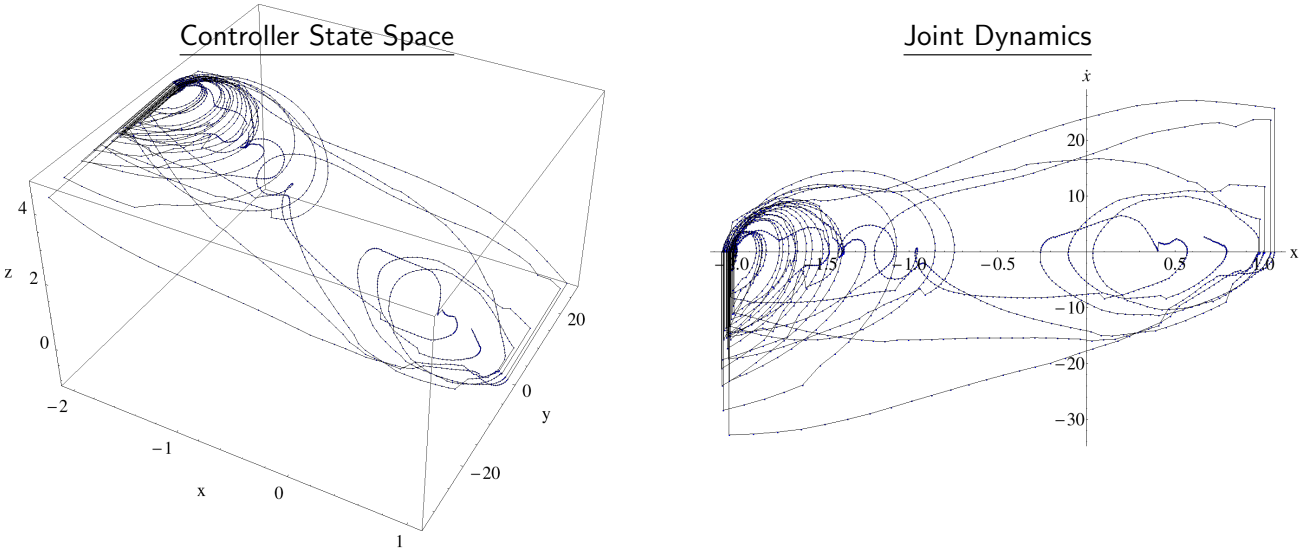


FIGURE 4.29: Dynamics of a chaotic controller and joint if the jumper is off ground. The trajectory on the chaotic controller does not show any prominent periodicity, however the joint limits at $-2$ and $1$ can be seen in the x dimension, because the trajectory hits them several times.

The same controller features a period 1 limit cycle due to the ground limiter, which gets apparent in the lower right corner of the plot. The ground limiter moved into the phase space of the controller to the extent that the phase space of the chaotic controller only permits a trajectory on a period 1 limit cycle.

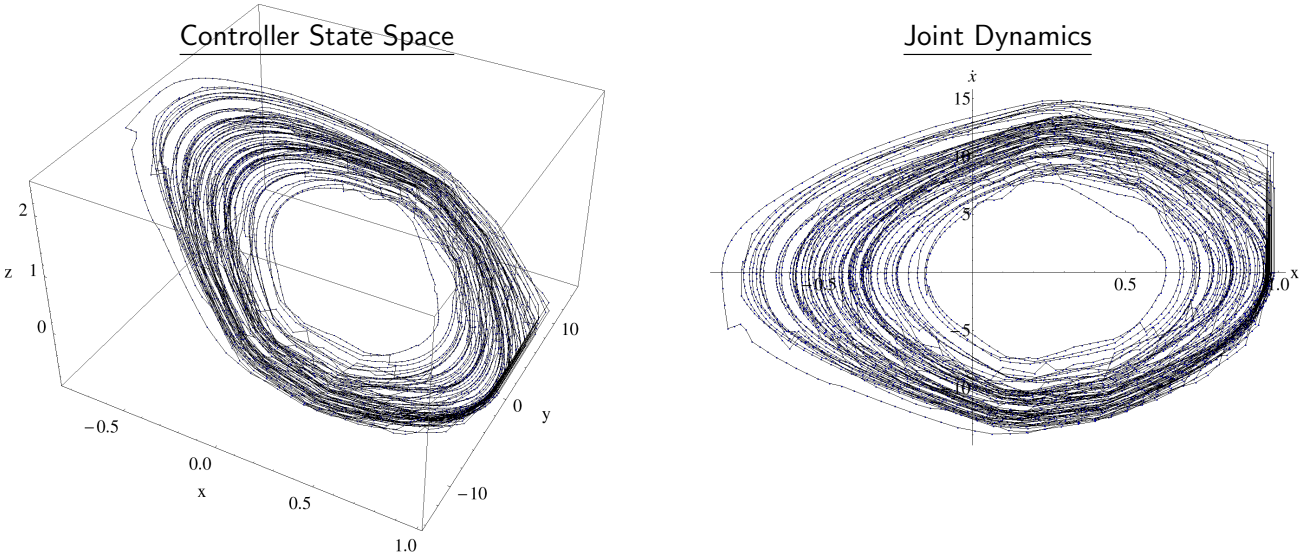


FIGURE 4.30: Dynamics of a chaotic controller and joint if the jumper is on ground. With the additional limiter of the ground, the dynamics of the controller change dramatically and quasi periodic limit cycle can be observed. The ground limit seems to have moved below 1, which stabilizes the trajectory on a period 1 limit cycle, hence making the creature locomote.

Conclusively it can be said that all three creatures show an adaption of behavior depending on whether they are on ground or off ground. However, the influence of the limiters are not always of the prominent nature of the simple limiter.

#### 4.2.2.3 Initial measure of gait stability

A simple measure of gait stability is the variability in the distance travelled by the creature when exposed to smaller perturbations. If the distance drops strongly when the gait is perturbed, the creature must have been strongly influenced by the perturbance (i.e. it has fallen over or has diverged from its initial path). To get a qualitative measure of stability, all controllers of the three locomoting creatures using directly limited, chaotic controllers (from section 4.2.2.2) were perturbed in the beginning of their evaluation. The perturbations are of small uniformly distributed zero mean nature and have an amplitude of $[0, 1 \cdot 10^{-8}]$. In the presence of such perturbations, it can easily be measured if the travelled distance varies. The perturbation experiment was repeated 100 times for each creature. For the three creatures, the locomoted distance varies less than $1\,\%$ over all runs. The controller is therefore stable enough to cope with smaller perturbations and returns to the original gait.

# Chapter 5

# Discussion and Outlook

An artificial evolution simulator is developed to evolve virtual three dimensional creatures built from rigid-body primitives. The simulator features the evaluation of one or multiple populations in different environmental and evolutional settings. Different fitness functions can be combined with weights and the fitness landscape can be changed during the evolutionary run. The simulations can be stopped, saved and continued later with little effort. The creatures can be manipulated during their evaluation and their internal controller and joint dynamics can be observed in real-time. The interface features various possibilities to log simulation data to file, so that further processing of the data is simplified.

Creatures were evolved using three different, distributed controller structures, a non-adaptive, sinusoidal controller and two newly introduced, adaptive chaotic system controllers. One chaotic controller is only indirectly limited through the morphology of the creature, the other directly limited through the sensory feedback in addition to the limits through the morphology. In order to develop the chaotic controllers, it was necessary to understand the dynamics to be expected when limiting a chaotic system with the limiter chaos control method. Therefore, the findings of the simple limiter chaos control method introduced in [2] with the example model of the Chua circuit [24] were confirmed and extended by the use of a tunable soft-limiter. Additionally, two different limiter configurations and their range of different periodic orbits were analyzed. It was found that a soft-limiter expands the size of each limiter parameter range of periodic orbit and therefore makes it easier to achieve stable orbits. The developed chaotic controller internally controls an underlying chaotic model system into chaotic and periodic trajectories using a variation of the simple limiter chaos control method. In the directly limited case, the simple limiter hereby is applied through a coupling with the controlled joint DoF by feeding the joint position and velocity output back into the chaotic controller, replacing the original state with the sensor values. Therefore, the directly limited, chaotic controller faces a limiter if the joint faces an external limiter

imposed through the environment. Limited by this external load, the chaotic system changes its internal dynamics to cope with the new load. The indirectly limited, chaotic controller does not have this sensory feedback and therefore always exerts a chaotic trajectory, which is then only influenced by the limits of the morphology. The evolutionary algorithm in this context composes different creatures to find a morphology and appropriate internal control parameters which exploit the internal chaotic dynamics of the controller in such a way that together they show basic locomotion behavior. For the sinusoidal and the directly limited, chaotic controller, creatures could be evolved showing various ways to locomote. More generally, it could be shown that using an arbitrarily chosen chaotic system and a simple limiter varying its limitation of the state space depending on the external constraints from the environment, different adaptive locomotion patterns can arise, which are robust to external disturbances. Using the indirectly limited, chaotic controller, no locomoting creatures could be evolved. This can be explained through the fact that the limits of the morphology did not influence the controller behavior, and therefore no periodic movement was exerted from the controller. Furthermore, for the directly limited, chaotic controller, it could be shown that different leg dynamics are emerging depending on the external load of the environment. This could be shown in a hand-made model organism as well as in several evolved creatures, that showed different leg dynamics when lifted off the ground than when left on ground to locomote. In experiments with the model leg, featuring one single joint connecting two body parts, different periodic orbits could be stabilized. The influence of different joint morphological parameters on the behavior of the model leg, namely the amount of joint torque and damping, could be observed. Applying higher or lower amounts of joint torque increases or decreases the orbit diameter respectively, damping leads to a smoother curvature of the orbit. However, in all cases the joint converges to a proper periodic orbit.

In the evolved creature case, only some creatures could be found to feature simple limiter control behavior, while others do not it very prominently. Still, given that stabilized orbits can be observed in some creatures makes it a valid hypothesis to be pursued further.

An issue confronted with during the simulation run was, that no fitness function accounted for gait efficiency, therefore, some creatures considered fit by the fitness functions in use did not perform an efficient locomotion gait. The simulation additionally has a general tendency to come up with very messy creature morphologies exploiting oscillations of large amplitude, which would require large amounts of energy to keep up the resulting locomotion. An improvement to address this issue would be that a creature had an amount of energy to spend proportional to its size or weight, so that a creature could not waste an excessive amount of energy on inefficient movements.

Another issue to address in the future would be the configuration of the limiter. The current simple limiter was evaluated experimentally to stabilize a periodic orbit, which is an indication

that the limiter is applied to the system. This could be addressed by a less restricting application of the sensory feedback, such as that a higher body part velocity would increase the size of the phase space of the chaotic system, so that dynamics of higher periodicity would arise. However, it is in question how to design the limiter, because as it is important to not directly build in a desired outcome. Therefore the choice of an appropriate limiter configuration is a particularly hard task.

A future experiment could be to actually measure the chaoticity of the controller and joint dynamics. Preliminary results suggest that the controller could be an intermittent system switching back and forth between chaotic and periodic phases. An experiment to measure the chaoticity would involve a successfully locomoting creature's controllers to be perturbed using a small perturbation. If the controller is stabilized on a periodic orbit, the trajectory then would be moved away from the orbit by the perturbation, but finally converge back onto the limit cycle. However, if the controller is not stabilized, the trajectory would diverge exponentially from the original, unperturbed trajectory and never return. The divergence of behavior could be measured using the Largest Lyapunov exponent, the mean rate of separation of trajectories, and calculated with the form adapted from [28] as described in Appendix A. The Largest Lyapunov exponent then expresses if the controller has an attracting, stable limit cycle to which it returns after having been slightly perturbed with uniformly distributed perturbances around zero. The perturbation can be automatically performed using the simulator by enabling automatic perturbation. The preliminary results, suggesting that the chaotic controller is an intermittent system, raise the question at which point during the evaluation the perturbation should be applied. A perturbation before the collision with the ground might be swallowed by the larger perturbation of the collision, obfuscating the actual results. Given that the creature starts above ground and falls onto it first, the best guess would be to perturb the creature after it has stabilized on ground and begun to perform its locomotion gait.

# Appendix A

The divergence of gait performance can be measured using the Largest Lyapunov exponent, the mean rate of separation of trajectories, and calculated with the following form adapted from [28]. Given the chaotic system in the form below, so that the iterated function applied $t$ times to $x_0$ results in $x(t)$, the same holds for a perturbance $\delta x_0$ which after t iterations occurs as $\delta x(t)$.

$$x(t) = f^t(x_0)$$
$$x(t) + \delta x(t) = f^t(x_0 + \delta x_0)$$

To measure the sensitivity to the initial conditions, the evolution of the perturbance $\delta x_0$ through the system can be calculated:

$$||\delta x(t)|| = |||\delta x_0||e^{\mu t}$$

The $\mu$ in that case is also called the Largest Lyapunov exponent (LLE). If $\mu > 0$, then the trajectory diverges exponentially. A $\mu = 0$ means, that the system is in a critical state, because its trajectories neither diverge nor converge. However, if $\mu < 0$, it expresses the convergence of a trajectory to a stable point or orbit. The quantification would be done by plotting the Log divergence distance ratio:

$$\log\left(\frac{||\delta x(t)||}{||\delta x_0||}\right) = \mu t$$

The LLE can then be found by line fitting segments of the time evolution log plot of the trajectory differences and then measuring the slope of the lines.

# Bibliography


[1] Rabinovich M. I., Abarbanel H. D. I., Huerta R., Elson R., and Selverston A. I. Self-regularization of chaos in neural systems: experimental and theoretical results. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **44**: 997–1005, 1997. DOI: 10.1109/81.633889

[2] Corron N. J., Pethel S. D., and Hopper B. A. Controlling chaos with simple limiters. *Phys. Rev. Lett.*, **84**: 3835–3838, 2000. DOI: 10.1103/PhysRevLett.84.3835 URL: http://link.aps.org/doi/10.1103/PhysRevLett.84.3835

[3] Grillner S., Cangiano L., Hu G.-Y., Thompson R., Hill R., and Warrén P. The intrinsic function of a motor system – from ion channels to networks and behavior. *Brain Research*, **886**: 224–236, 2000. DOI: http://dx.doi.org/10.1016/S0006-8993(00)03088-2 URL: http://www.sciencedirect.com/science/article/pii/S0006899300030882

[4] Hultborn H. and Nielsen J. B. Spinal control of locomotion – from cat to man. *Acta Physiologica*, **189**: 111–121, 2007. DOI: 10.1111/j.1748-1716.2006.01651.x URL: http://dx.doi.org/10.1111/j.1748-1716.2006.01651.x

[5] Hooper S. L. In: *Central pattern generators*. John Wiley & Sons, Ltd, 2001. DOI: 10.1038/npg.els.0000032 URL: http://dx.doi.org/10.1038/npg.els.0000032

[6] Brown T. G. The intrinsic factors in the act of progression in the mammal. *Proceedings of the Royal Society of London B: Biological Sciences*, **84**: 308–319, 1911. DOI: 10.1098/rspb.1911.0077 eprint: http://rspb.royalsocietypublishing.org/content/84/572/308.full.pdf URL: http://rspb.royalsocietypublishing.org/content/84/572/308

[7] Ayers J., Carpenter G. A., Currie S., and Kinch J. Which behavior does the lamprey central motor program mediate? *Science*, **221**: 1312–1314, 1983. DOI: 10.1126/science.6137060 eprint: http://science.sciencemag.org/content/221/4617/1312.full.pdf URL: http://science.sciencemag.org/content/221/4617/1312

[8] Harris-Warrick R. M. and Cohen A. H. Serotonin modulates the central pattern generator for locomotion in the isolated lamprey spinal cord. *Journal of Experimental Biology*, **116**: 27–46, 1985. eprint: http://jeb.biologists.org/content/116/1/27.full.pdf URL: http://jeb.biologists.org/content/116/1/27

[9] Ijspeert A. J. and Kodjabachian J. Evolution and development of a central pattern generator for the swimming of a lamprey. *Artif. Life*, **5**: 247–269, 1999. DOI: 10.1162/106454699568773 URL: http://dx.doi.org/10.1162/106454699568773

[10] Ijspeert A. J. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, **21**: Robotics and Neuroscience, 642–653, 2008. DOI: http://dx.doi.org/10.1016/j.neunet.2008.03.014 URL: http://www.sciencedirect.com/science/article/pii/S0893608008000804

[11] Poincaré H. Les méthodes nouvelles de la mécanique céleste. *Paris : Gauthier-Villars*, **3**: 434, 1892. URL: https://archive.org/details/lesmthodesnouv03poin

[12] Verhulst P.-F. Notice sur la loi que la population suit dans son accroissement. *Correspondance mathématique et physique*, **10**: 113–121, 1838. URL: http://books.google.com/?id=8GsEAAAAYAAJ&q=

[13] Lorenz E. N. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, **20**: 130–141, 1963. DOI: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2 URL: http://journals.ametsoc.org/doi/pdf/10.1175/1520-0469(1963)020%3C0130:DNF%3E2.0.CO;2

[14] Motter A E. and Campbell D. K. Chaos at fifty. *Physics Today*, **66**: 27, 2013. DOI: 10.1063/PT.3.1977 URL: http://scitation.aip.org/content/aip/magazine/physicstoday/article/66/5/10.1063/PT.3.1977

[15] Ott E., Grebogi C., and Yorke J. A. Controlling chaos. *Phys. Rev. Lett.*, **64**: 1196–1199, 1990. DOI: 10.1103/PhysRevLett.64.1196 URL: http://link.aps.org/doi/10.1103/PhysRevLett.64.1196

[16] Wagner C. and Stoop R. Optimized chaos control with simple limiters. *Phys. Rev. E*, **63**: 017201, 2000. DOI: 10.1103/PhysRevE.63.017201 URL: http://link.aps.org/doi/10.1103/PhysRevE.63.017201

[17] Ellenberger B. *Minemonics creature simulator*. Feb. 29, 2016. URL: https://github.com/benelot/minemonics

[18] Coumans E. *Bullet physics library*. Aug. 11, 2015. URL: http://bulletphysics.org

[19] Lee Graham. *Exaptation and Functional Shift in Evolutionary Computing*. AAINR47465. Ottawa, Ont., Canada, Canada, 2009. URL: https://curve.carleton.ca/system/files/etd/8b42f107-f3ee-40ab-9f4f-ac174969f74f/etd_pdf/5050c93842cbfe8f64fbcfa9108c736e/graham-exaptationandfunctionalshiftinevolutionary.pdf

[20] Coumans E. "Exploring MLCP solvers and featherstone" in: GDC '14 West Hall, Georgia, 2014. URL: http://box2d.org/files/GDC2014/ErwinCoumans_ExploringMLCPSolversAndFeatherstone.pdf

[21] *OGRE - Open Source 3D Graphics Engine*. Nov. 24, 2013. URL: http://www.ogre3d.org

[22] *CeGUI Graphics Library*. July 17, 2014. URL: http://cegui.org.uk/

[23] *Boost C++ Libraries*. Dec. 17, 2015. URL: http://www.boost.org

[24] Matsumoto T., Chua L., and Komuro M. The double scroll. *IEEE Transactions on Circuits and Systems*, **32**: 797–818, 1985. DOI: 10.1109/TCS.1985.1085791

[25] Devaney R. L. *An Introduction to chaotic dynamical systems*. 2nd Advanced book program Reading (Mass.): Addison-Wesley, 1989. URL: http://opac.inria.fr/record=b1077662

[26] Bendixson I. Sur les courbes définies par des équations différentielles. *Acta Mathematica*, **24**: 1–88, 1901. DOI: 10.1007/BF02403068 URL: http://dx.doi.org/10.1007/BF02403068

[27] Kennedy M. P. Three steps to chaos - Part 1:Evolution. *IEEE Trans. on Circuits and Systems*, **40**: 640, 1993. DOI: 10.1109/81.246140 URL: http://www.eecs.berkeley.edu/~chua/papers/Kennedy93.pdf

[28] Rosenstein M. T., Collins J.J., and De Luca C. J. A practical method for calculating largest Lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, **65**: 117–134, 1993. DOI: http://dx.doi.org/10.1016/0167-2789(93)90009-P URL: http://www.sciencedirect.com/science/article/pii/016727899390009P