GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Bachelorarbeit

Limit Cycles of neuron controlled robots in simulated physical environment

Tim Jahn

Institut für Theoretische Physik
Johann Wolfgang Goethe-Universität
Frankfurt am Main

Mai 2015

Erstprüfer:	Prof. Dr. Claudius Gros
Zweitprüfer:	Prof. Dr. Roser Valentí

Contents

1 Introduction

Humans have been fascinated by the idea of building robots and creating artificial intelligence since the appereance of the first computers. In the early times engineers and scientists were aiming to build machines controlled by a computer program, which acts autonomously beside humans.
Although the field of robotics has made huge progress in solving many specific problems, such as the automatisation of industrial fabrication processes or controlling simple humanoid robots or drones, the world is far too complicated to have a comprehensive model of everything around us in a single conventional computer program. An alternative approach is to go one step back and not to develop a brain, which already has a model for the world, but to create a model, following general principles, which would make the robot able to adapt to the world. From these general principles one derives the equation of motions, the time evolution of the internal variables. In the following we will discuss a few different suggestions. One principle could be, to act in a way which maximises future options. Another idea is, to minimise the prediction error of the effects of one's actions. A third one is, to maximise the information (entropy) of some characteristic distributions of the robot's internal variables.
In the first case one can think about a man in a labyrinth. At every branch-off he chooses the direction wich has the most outgoing paths.
The second case is discussed intensively in the book entitled 'The Playful Machine' ([DM11]). Here one might consider a bee flying against a window. While it cannot see the barrier, it feels its effect, which yields to an error between its own and the expected position. As a consequence it tries to act differently by changing the direction.
This thesis sticks to the third principle. A simple robot with limited knowledge about itself and the surrounding world is controlled by a neuron, which adapts its normalised firing rate (the activity distribution) to maximise the entropy of this distribution. The dynamics of one neuron has already been examined by Walter ([Wal13]), but without feedback coming from the environment. This robot with the one neuron is simulated in a physical environment, called LPZROBOTS, developed by the group 'Research Network for Self-Organization of Robot Behavour' in Leipzig.
First a small overview of the software and its various optional tools is given. The examined robot, controller and motors are discussed in detail. This is followed by an analysis of the idealised underlying dynamical system, for which the fixpoints, their stability and the limit cycles are determined.
In the main part the different behaviours of the robot thrown in this environment are described and compared. What is essential is the way in which the change of the adaption parameters yields to specific behaviours. Additionaly a small insight of an extended system is given in which a second neuron is added and their interaction leading to self-organisation is shortly shown.
Finally a summary of results with an outlook of possible future research options is given.

2 Theory and Setup

2.1 The LPZROBOTS Simulator

LPZROBOTS is a robot simulator with a graphical interface implemented in C++ and developed by the group ‘Research Network for Self-Organization of Robot Behavior’ (Leipzig/Göttingen/Edinburgh) and based on the widely used software library ODE (Open Dynamic Engine). The simulated world consists of simple objects with a fixed geometric shape (like cylinders, spheres, boxes etc.), which can be connected through different types of joints (slider joints, hidge joints etc.) to form robots. One can use materials with different properties (elasticity, hardness etc.) for the objects. The semi-implicit Euler method is used for integrating the differential equations ([DM11]). Each object follows the Newtonian laws of physics, namely gravitation, friction, centrifigual forces and others. The simulator combines many useful features. It is possible to change parameters online from the command line. The Guilogger, an external tool, allows one to plot the parameters and variables while running the simulation, which alternatively can be visualised by Matrixviz. Additionally one can overload different functions and define custom keyboard shortcuts to control the simulation process. For example, forces and torques can be applied to the robot manually. This feature was used intensively to generate certain dynamics. A typical simulation is built up as follows: the user defines the frame conditions of the virtual world by setting up the environment and initialises the obstacles (passive objects) and the agents (controlled robots). After the intialisation the simulator enters the main loop and performs iteratively physical simulation steps using the ODE. The update of the graphics is done by using ‘Open Scene Graph’ with a frame rate of 25 fps ([DM11]).

A robot basically consists of three parts: A body with some geometrical shape, motors and sensors. The robot can just gain information about its own state and the environment through sensors (for example the position of some internal state or the actual velocity). With its motors the robot can perform certain actions; there are for example angular motors accelerating wheels or slider servos, which can move a weight on an axis.

Every robot comes along with a controller, the ‘brain’ of the robot, which is in principle a function, mapping the current sensor values to motor commands. The parameters defining the properties of this function can be adapted, making learning possible. An important but simple controller without learning is the ‘sinecontroller’ which sends motorcommands according to a sine function.

2.2 Barrel with one bar

The robot (Fig. 2.1) mainly examined in this thesis is closely related to the Barrel2Masses and the SphereRobot3Masses built-in robot classes of LPZROBOTS. It is a cylinder with

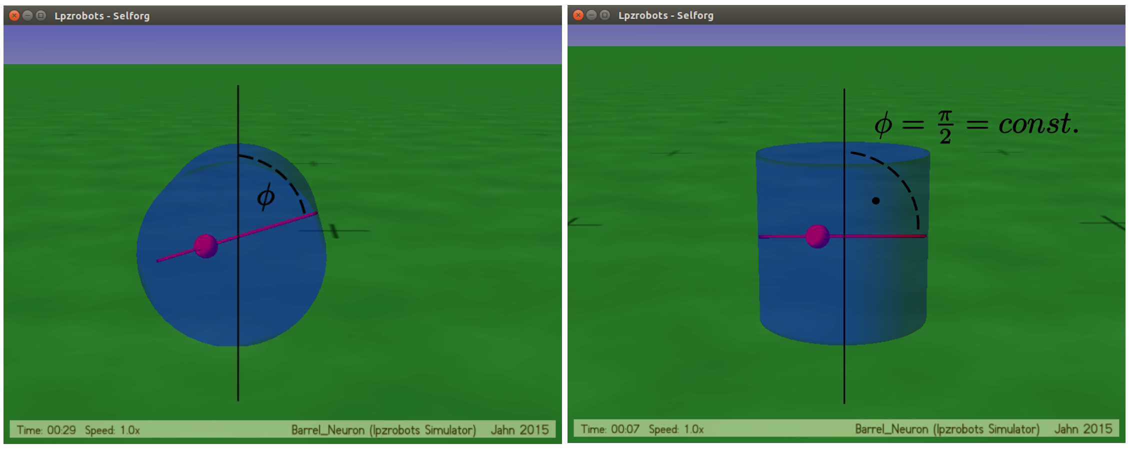

Lpzrobots - Selforg
ϕ
Time: 00:29 Speed: 1.0x
Barrel_Neuron (lpzrobots Simulator) Jahn 2015
Lpzrobots - Selforg
$\phi = \frac{\pi}{2} = const.$
Time: 00:07 Speed: 1.0x
Barrel_Neuron (lpzrobots Simulator) Jahn 2015

Figure 2.1: The robot which is examined in this thesis. The main body is a cylinder with one internal axis on which a pendular is moved by a servo, the robot starts moving due to the change of the barycenter. The black line and ϕ denote the global z-axis and the angle between it and the internal axis. *left*: The barrel in rolling motion. By construction it is (theoretically) just possible to move along the global x-axis. *right*: The barrel is placed on its side. In this position the influence of the environment is minimised.

an internal axis fixed orthogonal to the z-axis of the cylinder, with a small spherical mass on it, called the pendular, which can be moved by a slider servo. Moving of the small mass leads to a change of the centre of mass. The robot's only sensor measures the actual position of the small sphere on the axis, normalised to [-1,1]. Theoretically this robot should just be able to move along the x-axis, but unfortunately for some parameter configurations, strange behaviours occur. This will be shortly discussed in the appendix. The dynamics of the robot is regulated by a controller following learning rules according to [LG13], which are described in more detail in section 2.3.2 . The system is inspired from the single self-coupled neuron called autapse.

The motors implemented in the simulation are realistic, hence they have finite power and work according to the physical laws. After receiving a command from the controller the motor does not make the pendular jump instantly to the desired position, but a force or torque takes effect which should drive it to this state. This force/torque is calculated using proportional-integral-derivative control (PID control), a general and widely used mechanism from feedback and control theory.

2.2.1 Feedback and Control Theory

Consider a simple system characterised by a measurable variable, S. The system is subject to the variation of some unknown environmental variable ν. Additionally one is able to modify the state of the system, S, with some control input, u. Therefore the system has the functional form $S = S(u, \nu, t)$, where t denotes the time variable. The goal is to lock the state, S, of the system to some value, S_t, and keep it there, regardless of the variation of ν. To reach this, S and the difference (error), $e = S - S_t$ is measured. With this error a control value, $u(e)$, is calculated and applied to the system. This procedure is repeated in every timestep. Depending on the problem, one has different possibilities to calculate the feedback control variable $u(e)$.

2.2.1.1 PID controller

An easy and effective way is to use the following feedback gain:

$$u(e,t) = g_P e(t) + g_I \int_{t_0}^{t} e(\tau)\mathrm{d}\tau + g_D \frac{d}{dt} e(t) = P + I + D \tag{2.1}$$

where g_P, g_I and g_D are the gain rates which have to be tuned carefully. Roughly speaking, one calculates u taking the present error (P), the accumulated past errors (I) and a prediction of future errors (D) based on its current rate of change. This method can be generalised to more system/control variables. In the case of the robots examined in this thesis there is the following situation: the system is characterised by the actual position, x_a, of the pendular on the bar and the desired target position, x_t. For simplicity in the following x_a will often be denoted by x, hence the measurable variable S of the controller is equal to x. The slider servo can add a force to this sphere of mass m, $F = m\ddot{x} = u$. So Eq. (2.1) looks like

$$m\frac{\mathrm{d}^2}{\mathrm{d}t^2}x(t) = g_P(x_t - x(t)) + g_I \int_{t_0}^{t} (x_t - x(\tau))\mathrm{d}\tau + g_D \frac{\mathrm{d}}{\mathrm{d}t}(x_t - x(t)) \ . \tag{2.2}$$

Setting $g_I = 0$ and doing some algebra, this yields to

$$(x_t - x(t)) + \frac{g_D}{g_P}\frac{\mathrm{d}}{\mathrm{d}t}(x_t - x(t)) + \frac{m}{g_P}\frac{\mathrm{d}^2}{\mathrm{d}t^2}(x_t - x(t)) = 0 \ , \tag{2.3}$$

which is the differential equation of a damped spring with an elongation of $\Delta x = x_t - x(t)$. Compare this to the movements of extremities in nature. In LPZROBOTS the PID controller is implemented in the following way:

$$m\frac{\mathrm{d}^2}{\mathrm{d}t^2}x = K_P\left[(x_t - x) + K_D\frac{\mathrm{d}}{\mathrm{d}t}(x_t - x)\right] \tag{2.4}$$

where $K_P = g_P$, and $K_D = \frac{g_D}{g_P}$. To verify that the slider servo works as expected, the equation of the PID controller is solved numerically and compared with the data of the simulation. To do this Eq. (2.4) is transfered into a system of first order equations :

$$\begin{aligned} \dot{x} &=: v \\ \dot{v} = \ddot{x} &= \frac{K_P}{m}[(x_t - x) - K_D(v - \dot{x}_t)] \end{aligned} \tag{2.5}$$

The solution is calculated using a simple Euler method with stepsize $\mathrm{d}t = 0.01$

$$\begin{aligned} v(t+\mathrm{d}t) &= v(t) + \dot{v}(t)\mathrm{d}t \\ x(t+\mathrm{d}t) &= x(t) + \dot{x}(t)\mathrm{d}t = x(t) + v(t)\mathrm{d}t \ , \end{aligned} \tag{2.6}$$

with the initial conditions:

$$x(t_0) = v(t_0) = 0 \ . \tag{2.7}$$

The barrel is placed on top (see Fig. 2.1), such that the axis is oriented parallel to the ground. Hence the only force acting on the pendular is the one generated by the slider servo.
The figures 2.2 and 2.3 show that a relatively good matching is obtained, and hence that the motor works adequately.

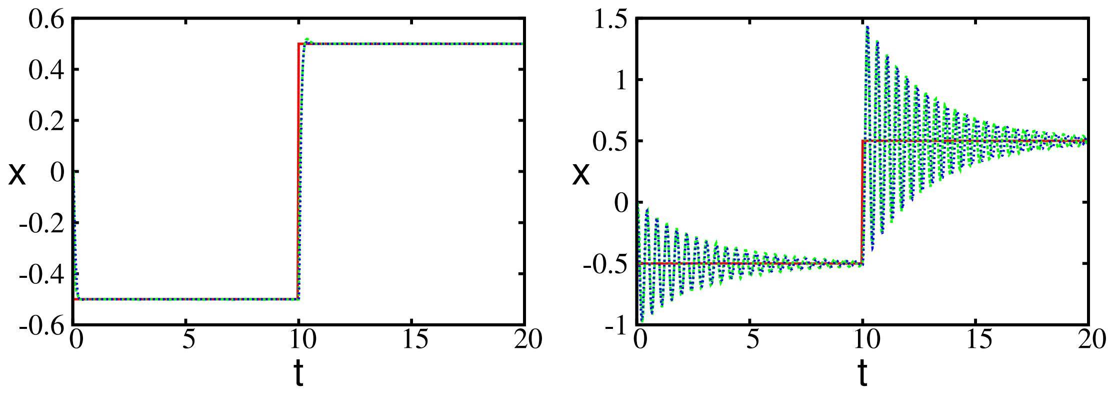


Figure 2.2: Testing of the PID controller; Red: controller output (target position x_t), green: sensor value (actual position x_a), blue: numerical solution of Eq. (2.2.1.1) with $\mathrm{d}t = 0.01$. The green and blue curves fall on each other. *Left*: The parameters used for the simulation, close to critical damping ($K_P = 200$, $K_D = 0.1$, $K_I = 0$). *Right*: Nearly undamped case. Note that for this figure the jointbounds have been increased, hence the pendular can overshoot the $[-1, 1]$ interval ($K_P = 200$, $K_D = 0.035$, $K_I = 0$). The stepsize is $\mathrm{d}t = 0.01$.

2.2.1.2 Critical damping

In the case of $g_I = 0$ (also called PD control) and given g_P, a good choice is to set g_D to critical damping g_D^c. The damped harmonic oscillator

$$\ddot{x} + a\dot{x} + bx = 0 \ , \tag{2.8}$$

where a is the damping and $\sqrt{b}$ is the natural frequency has the general solution

$$x(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} \ , \tag{2.9}$$

where $\lambda_{1/2} = -\frac{a}{2} \pm \sqrt{\frac{a^2}{4} - b}$ are the eigenvalues of the oscillation. The oscillator is critically damped if there is just one degenerated eigenvalue. Hence the condition is given by

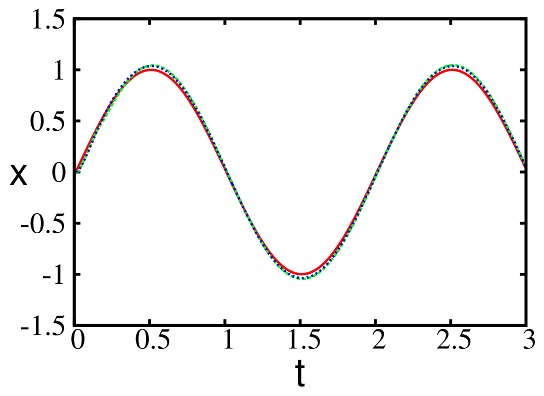
$$a^2 = 4b \ . \tag{2.10}$$

In case of LPZROBOTS we have ($e = (x_t - x)$):

$$m\ddot{e} = -K_P(e + K_D\dot{e}) \tag{2.11}$$

$$\Rightarrow \ddot{e} + \frac{K_P K_D}{m}\dot{e} + \frac{K_P}{m}e = 0 \tag{2.12}$$

Hence $K_D^c = 2\sqrt{\frac{m}{K_P}}$. The parameters for the main simulation in this thesis are $K_P = 200$, $K_D = 0.1$ and $m = 1$ while $K_D^c = 2\sqrt{\frac{K_P}{m}} = \frac{\sqrt{2}}{10} \approx 0.1414$. The damping is slightly subcritical.



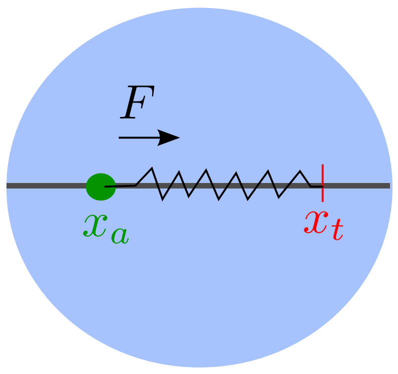


Figure 2.3: *Left*: The target position x_t is changing according to a sine function; Red: controller output (target position x_t), green: sensor value (actual position x_a), blue: numerical solution of Eq. (2.2.1.1) with d$t = 0.01$. The green and blue curves fall on each other. The fitting is even better for continuous contoller commands. *Right*: Illustration of how the motor works. The force acting on the pendular is calculated according to a damped spring with centrepoint x_t.

2.3 Neural dynamics

2.3.1 General principles

As a model of a neuron, two states are considered; that of being at rest or that of firing. The state of a neuron depends on its membrane potential, $x(t)$. Once the membrane potential reaches a certain threshold the neuron generates a spike, then the membrane potential returns rapidly to the resting mode. Averaging the firing events over time, one gets a normalised firing rate, $y(x(t)) \in [0, 1]$. The time average y is hence dependent on the membrane potential x which changes according to some input. The firing rate statistics is given by

$$p(z) = \frac{1}{T}\int_{t_0}^{t_0+T} \delta(z - y(t))\mathrm{d}t, \qquad \int_0^1 p(z)\mathrm{d}z = 1 \tag{2.13}$$

where t_0 is the initial time and T the measured time. T would be substantially larger than the time that one firing event takes. A general question arises as to how the firing rate distribution $p(y)$ looks like. Here two underlying general principles are considered; on the one hand the system aims to minimise the energy used, while on the other hand attempts to maximise its output information entropy [SK99]. Where the expectation value, μ, and the standard deviation, σ, are fixed, the entropy of a distribution is maximised by a Gaussian probability distribution [Gro10]:

$$q(y) \propto \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) \propto \exp(\lambda_1 y + \lambda_2 y^2) \tag{2.14}$$

Following these principles the goal is to minimise the Kullback-Leibler divergence [Gro10] of $p(y)$ and $q(y)$:

$$D_{KL}(p, q) = \int p(y) \ln \frac{p(y)}{q(y)} \mathrm{d}y \tag{2.15}$$

2.3.2 One neuron with internal adaptation

In this thesis a neuron is considered, which controls the dynamics of the robot and is coupled to itself through the environment. Let x_a denote the position of the small sphere on the bar. The membrane potential is assumed to be a linear function of x_a and the firing rate of the neuron is given by the transfer function used by Linkerhand & Gros [LG13]

$$y(x_a) = \frac{1}{1 + e^{-a(x_a - b)}} \ , \tag{2.16}$$

where $a(t) > 0$ and $b(t)$ are the adapting gain and threshold parameters. The target position is given by:

$$x_t(y(x_a)) = 2\left(y(x_a) - \frac{1}{2}\right) = f(x_a) \ , \tag{2.17}$$

which is just the output firing rate, rescaled to $[-1, 1]$. The transformation is recommended to make self-coupling possible, so the range of x_t should be equal to the domain of the input x_a. The target function q is desired to be symmetric and centred around the origin, hence $\lambda_1 = -\lambda_2 = \lambda$ are chosen. The equations of motion for the internal parameters a and b are derived using the second principle, the maximisation of entropy. Minimising D_{KL} in terms of a and b leads to the following equations [LG13]:

$$\begin{aligned} \dot{a} &= \epsilon_a \left(\frac{1}{a} + (x - b)\theta\right) \\ \dot{b} &= -\epsilon_b a \theta \end{aligned} \tag{2.18}$$

where $\theta = 1 - 2y + (-\lambda + 2\lambda y)(1 - y)y$ and ϵ_a, ϵ_b are appropriately chosen adaptation rates (see section 2.4).

In contrast to the autapse [Wal13], y or respectively x_t is not the input for the next time step, but $x_t(y)$ is given as a motor command to the PID controller. The input for the neuron in the next time step is then the new position, $x_a(t + dt)$, of the pendular, after the force computed by the PID controller was applied.

2.4 Timescales

Every part of a physical system evolves on different timescales. In the case of the neuron this could be the relaxation time after firing. Matching the timescales of the different parts can be an arduous task. While choosing values for ϵ_a and ϵ_b one has to take this into account. It has to match the rolling motion of the cylinder and the response time of the motor. Picking a very big value should theoretically lead to a quick convergence of the firing rate p to the target distribution q. In such a situation, the motor would be unable to follow the rapid changes of the controller commands and the dynamics repertoire of the robot is heavily reduced.

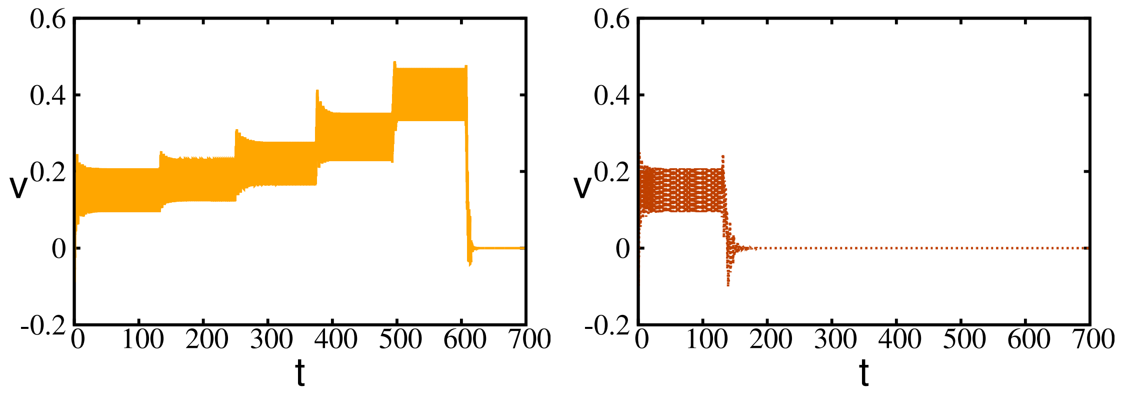


Figure 2.4: The barrel is equipped with the sine controller. The actual speed is plotted against the time, while the period T of the controller's sine function is decreased at certain moments. *Left*: The period is decreased slowly, the barrel accelerates. At some point ($T \in [20, 25]$) the robot cannot follow the rapidly changing commands of the controller anymore and the barrel stops. *Right*: The period is decreased directly from 60 to 30, a value for which the barrel was still in a smooth rolling mode in the case of slow decrease. The barrel stops immediately, hence the state of the system depends on the preceding events.

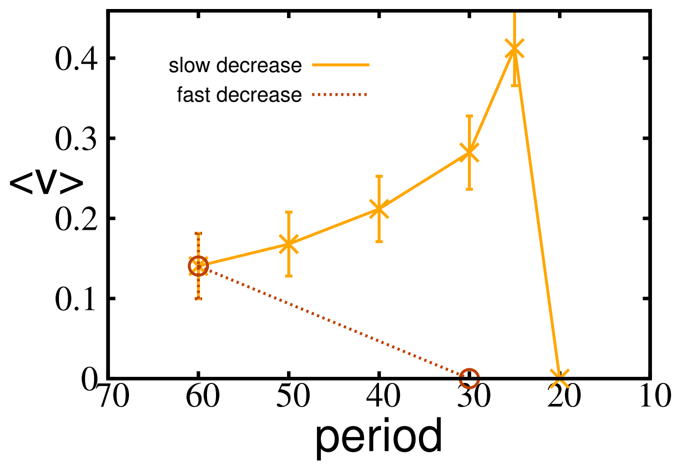


Figure 2.5: The average speed with standard deviation is plotted against the period. The crosses belong to the left plot of Fig. 2.4 (slow decrease), the circles to the right plot (rapid decrease). Note that the x-axis is inverted.

The robot is now equipped with the sine controller, which is an elementary tool to produce rolling motions, but it does not allow learning.

The controller generates commands according to a sine function. Starting with a period of 60 this controller leads the barrel to a smooth rolling motion. As the period of the sine is decreased the barrel becomes faster. At some point, depending also on the actual dynamics, this behaviour collapses and the motors cannot follow the rapid changes of the commands anymore.

In the right picture of Fig. 2.4 the period is setted directly to 30, the system collapses.

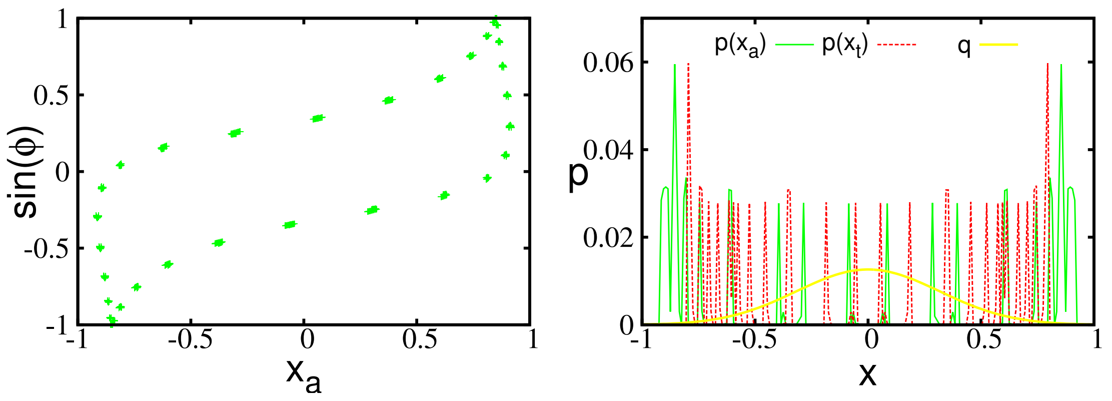


Figure 2.6: Coherence effects for $\epsilon_a = 2$, $\epsilon_b = 2.95$ and $\lambda = -5$ (compare Fig. A.2 second row). The simulation step size $\mathrm{d}t = 0.01$ and the timescale (the period) of the rolling motion are close to a rational relation. While it is a smooth motion, the system is measured in nearly the same states leading to 'confusing' data sets. All measured values concentrate around some discrete points. *Left*: The z-projection $\sin(\phi)$ of the axis against the position x_a of the pendular, compare to Fig. 2.1. *Right*: The distributions of x_a the actual position (green), x_t the controller output (red) and the Gaussian target distribution q (yellow). For the statistics 43500 data points are binned (200 bins).

But when decreasing the period slower (right picture of Fig. 2.4) the barrel rolls stable with a period even smaller then 25.

2.4.1 Coherence

Since the state of the system can be measured just at discrete points, it may happen that the period of some motion is a multiple of the simulation time step. In this case one always gets the same information, the plots of the data look highly regular while in the graphical visualisation the movement seems to be smooth. Here is an example: Fig. 2.6 belongs to the same motion like the second row of Fig. A.2 (just with a different ϵ_b), but one would interpret it as them belonging to some discontinuous motion where the pendular 'jumps'.

Note that the peaks of the right plot of Fig. 2.6 become more dense where the maxima of the right plot of Fig. A.2, second row, is located. Comparing the phase diagram with the simulation it becomes clear, that here coherent annihilation occurs.

3 Results

In the following the results of the computer simulations are presented. The exact parameters of the robot and the environment are listed in the Appendix.

3.1 Adiabatic fixpoint dynamics

First, the robot is placed on top (see Fig. 2.1 left plot), to minimise the influence of the environment. Gravitation and centrifugal forces have no effect then. For a fixed value $a = 2$ the system's dynamics are determined by the actual position, $x := x_a$, and velocity, v, of the pendular plus the adapting parameter b of the transfer function:

$$\begin{aligned} \dot{x}(t) &= v(t) \\ \dot{v}(t) &= \frac{K_P}{m}[(x_t(t) - x(t)) - K_D(v - \dot{x}_t(t))] \\ \dot{b}(t) &= -\epsilon_b a(1 - 2y + (-\lambda + 2\lambda y)(1 - y)y) \ . \end{aligned} \tag{3.1}$$

For a fixed value b the fixpoints and stability of the (x, v) sub-system are calculated. These fixpoints are called adiabatic fixpoints of the whole system, in analogy to adiabatic changes of states in thermodynamics (b is changing slowly). Note that, since $x_t(t) = x_t(x(t)) = 2\left(\frac{1}{1+e^{a(b-x(t))}} - \frac{1}{2}\right)$ it follows that $\dot{x_t}(t) = \frac{\partial x_t}{\partial x}\dot{x} = \frac{\partial x_t}{\partial x}v$. Hence, for an adiabatic fixpoint:

$$\begin{aligned} \dot{x}(t) &= 0 \quad \Rightarrow \quad v^* = 0 \\ \dot{v}(t) &= 0 \end{aligned} \tag{3.2}$$

$$\begin{aligned} \dot{v}(t) &= \frac{K_P}{m}[(x_t(t) - x(t)) - K_D(v - \dot{x}_t(t))] \\ &= \frac{K_P}{m}\left[(x_t(t) - x(t)) - K_D(1 - \frac{\partial x_t}{\partial x})v\right] = 0 \end{aligned} \tag{3.3}$$

$$\begin{aligned} &\Rightarrow x_t^*(t) - x^*(t) = 0 \\ &\Rightarrow x_t^* = x^* = f(x^*) = \left(\frac{1}{1 + e^{-a(x^*-b)}} - \frac{1}{2}\right) \ , \end{aligned} \tag{3.4}$$

since $v^* = 0$.

This is in accordance with what one would expect from a controller, i.e.: if the target position x_t is equal to the actual position x_a, no further action is needed. The second fixpoint condition cannot be solved in a closed form. However it is possible to express $b(x^*)$:

$$\begin{aligned} b(x^*) &= \frac{1}{a}\ln\left(\frac{2}{1 + x^*} - 1\right) + x^* \\ b'(x^*) &= 1 - \frac{2}{a}\frac{1}{2(1 + x^*) - (1 + x^*)^2} \ . \end{aligned} \tag{3.5}$$

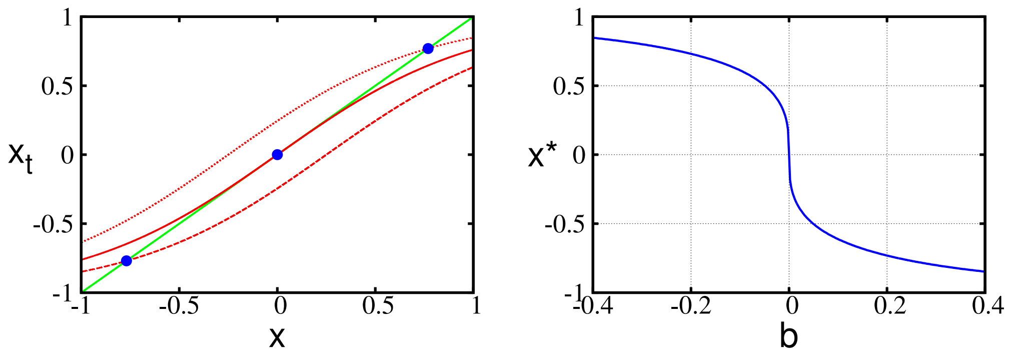


Figure 3.1: Fixpoint analysis of the (x, v) sub-system for $a = 2$ and constant b. Since the slope of the transfer function is $\frac{d}{dx}x_t = 2ay(1-y) \leq 2 \cdot 2 \cdot \frac{1}{2} \cdot \frac{1}{2} = 1 = \frac{d}{dx}x$ there is exactly one intersection. *Left*: The green and red curves denote the identity and the transfer function for different values of b (thick: $b = 0$, dotted: $b = -0.25$, dashed: $b = 0.25$) respectively. The blue dots denote the adiabatic fixpoints for different values of b. *Right*: The numerical solution of Eq. (3.4) with an accuracy of 0.001. Since the slope is very high at 0 (it is $\frac{1}{b'(0)} = 1 - \frac{2}{a} = -\infty$ from Eq. (3.5)), small perturbations of b lead to large changes of the adiabatic fixpoint's position. This, combined with the time lag of the system between x_t and x_a is what finally drives the dynamics.

Additionally a graphical analysis offers some insight (see Fig. 3.1, the intersecions of the graphs correspond to fixpoints of the system).

To determine the stability of an adiabatic fixpoint (x^*, v^*) the eigenvalues of the Jacobian are calculated

$$\begin{aligned}\mathcal{J}(x^*, v^*) &= \begin{pmatrix} 0 & 1 \\ \frac{K_P}{m}(-1+z) + K_D \cdot v \cdot \frac{\partial z}{\partial x} & -K_D(1-z) \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 \\ \frac{K_P}{m}(-1+z^*) & -K_D(1-z^*) \end{pmatrix} ,\end{aligned} \tag{3.6}$$

where $z^* = \frac{\partial x_t}{\partial x}_{x^*,v^*} = 2ay^*(1-y^*)$ and $y^* = y(x^*)$. The obtained eigenvalues are given by

$$\lambda_{1/2} = \frac{\mathrm{tr}(\mathcal{J}^*) \pm \sqrt{\mathrm{tr}(\mathcal{J}^*)^2 - 4 \cdot \det(\mathcal{J}^*)}}{2} , \tag{3.7}$$

where $\mathrm{tr}(\mathcal{J}^*)$ and $\det(\mathcal{J}^*)$ are the trace and determinant of $\mathcal{J}^* = \mathcal{J}(x^*, v^*)$. The signs of the real parts of the eigenvalues determine the stability of the fixpoint.

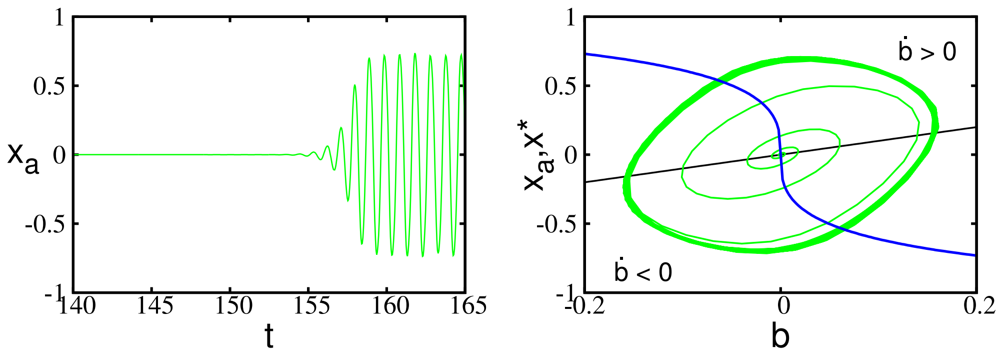


Figure 3.2: Examination of the fixpoint $(x^* = 0, v^* = 0, b^* = 0)$ of Eq. (3.1). At $t = 147$ a small perturbation is added to b ($\delta b = 10^{-7}$). The system quickly converges to the limit cycle. *Left*: The oscillation as a function of time. *Right*: The green and blue curves denote the phase space plot projected to the (x, b) plane and the adiabatic fixpoint $x*(b)$ respectively. Above the black line $\dot{b} > 0$, below $\dot{b} < 0$.

$$\begin{aligned}
\det(\mathcal{J}^*) &= 0 \cdot (-K_D(1-z^*)) - \frac{K_P}{m}(-1+z^*)) \cdot 1 \\
&= \frac{K_P}{m}(1-z^*) \\
&\geq \frac{K_P}{m}\left(1 - 2 \cdot 2 \cdot \frac{1}{2} \cdot \frac{1}{2}\right) \\
&= 0
\end{aligned} \tag{3.8}$$

$$\begin{aligned}
\mathrm{tr}(\mathcal{J}^*) &= 0 - K_D(1 - z*) \\
&\leq -K_D(1 - 2 \cdot 2 \cdot \frac{1}{2} \cdot \frac{1}{2}) \\
&= 0 \ ,
\end{aligned}$$

since $z^* = 2ay^*(1-y^*)$ and $y^*(1-y^*) \leq \frac{1}{2} \cdot \frac{1}{2}$. Note that for $K_D = 0.1$ and $\frac{K_P}{m} = 200$ the argument of the squareroot is negative. Hence

$$\Re(\lambda_{1/2}) = \mathrm{tr}(\mathcal{J}^*) \leq 0 \ , \tag{3.9}$$

with equality for $y^* = \frac{1}{2}$ ($\Rightarrow x^* = 0$). The adiabatic fixpoint is stable for all $x \neq 0$.

Now the dynamics of b is taken into account.

$$\begin{aligned}
\dot{b} &= -\epsilon_b a(1 - 2y + (-\lambda + 2\lambda y)(1-y)y) \\
&= \epsilon_b a(2y-1)(1 - \lambda(1-y)y)
\end{aligned} \tag{3.10}$$

Since $y \in [0, 1]$, for $\lambda < 0$ the sign of $\dot{b}$ is determined by the sign of $(2y - 1)$, which is determined by the relation between x and b

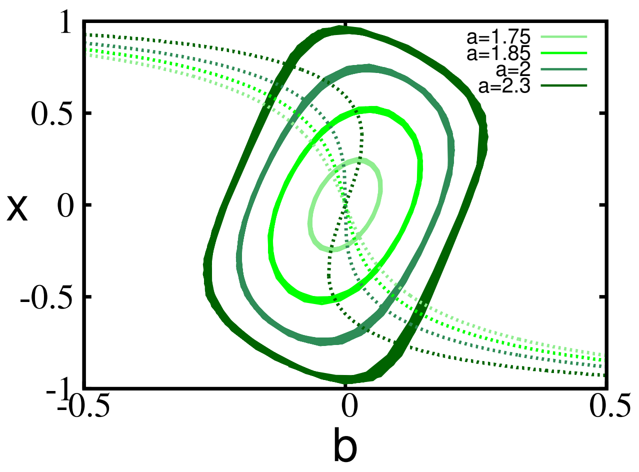


Figure 3.3: Phase plane plots of the limit cycle. The dotted lines denote $x^*(a,b)$. The critical point of the Hopf-bifurcation is around $a \approx 1.7$, compared to $a = 2$ in case of the autapse [Wal13].

$$\dot{b} \begin{cases} < 0, & \text{for } x < b \\ = 0, & \text{for } x = b \\ > 0, & \text{for } x > b \end{cases} \tag{3.11}$$

If $x < b$ then $\dot{b} < 0$ and hence $\dot{x}^* > 0$ (compare Fig. 3.1) and vice versa. For $a = 2$, $\lambda = -25$ and $\epsilon_b = 0.7$ this leads to an oscillation.

The fixpoint of the (3.1) system (x, v, b) is determined as follows:

$$\begin{aligned} \dot{x} = 0 \Rightarrow &\quad v^* = 0 \\ \dot{b} = 0 \Rightarrow &\quad b^* = x^* \\ \dot{v} = 0 \Rightarrow &\quad x^* = x_t^* \quad = f(x^*) = 0 \ , \end{aligned} \tag{3.12}$$

thus the only solution is $(x^* = 0, v^* = 0, b^* = 0)$.

The simulations show that this is an unstable fixpoint and since the system is bounded there has to be a limit cycle. Fig. 3.2 shows how the system converges to the limit cycle if the unstable fixpoint is perturbed.
This limit cycle can be viewed in the following way: the phase point tries to go to the stable adiabatic fixpoint, but the adapting threshold parameter, b, always moves that away, so that it can never actually reach it (compare to the so called 'Carrot and Donkey Dynamics' [GLW14]).

The structure of the limit cycle depends on a. For smaller values of a, its radius decreases until it vanishes by a Hopf-bifurcation at around $a = 1.7$, while for bigger values of a the radius increases (see Fig. (3.3)). In the following the simulations are done with $a = 2$.

If a simulation is started away from the fixpoint, the pendular quickly joins an oscillation motion, the limit cycle of the system. Calculating in every timestep the adiabatic fixpoint, one clearly sees how the motion is driven by this stable adiabatic fixpoint and its rapid motion when b passes 0 (see Fig. 3.4).

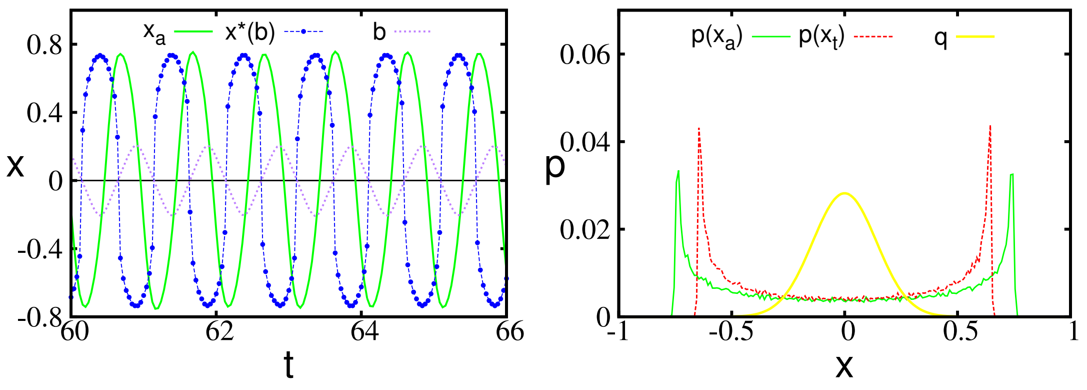


Figure 3.4: Examination of the dynamics with negligible influence of the environment (standing barrel of Fig. 2.1). *Left*: The motion of the small pendular. The green curve denotes the actual position x_a, the blue dots show the adiabatic fixpoint x^*. *Right*: The measured probability distributions (*red*: target position x_t, *green*: actual position x_a), differ from the Gaussian target distribution (*yellow*). For the statistics 43500 data points are binned (200 bins).

3.2 Barrel with one internal axis

To investigate the different rolling modes of the barrel, the robot is placed on the ground, its z-axis being parallel to it. The phase space gains two extra dimensions by the orientation of the internal axis, namely the angle ϕ of the internal axis relative to the ground and its angular velocity $\dot{\phi}$ (see Fig. 2.1). Hence the phase space $(x,v,b,\phi,\dot{\phi})$ is five dimensional, the dynamics of the system is investigated by numerical simulations. The behaviour of the robot (a periodic motion of the robot corresponds to a limit cycle of the underlying dynamical system), at least to some extent, can still be explained with the results of the previous section.

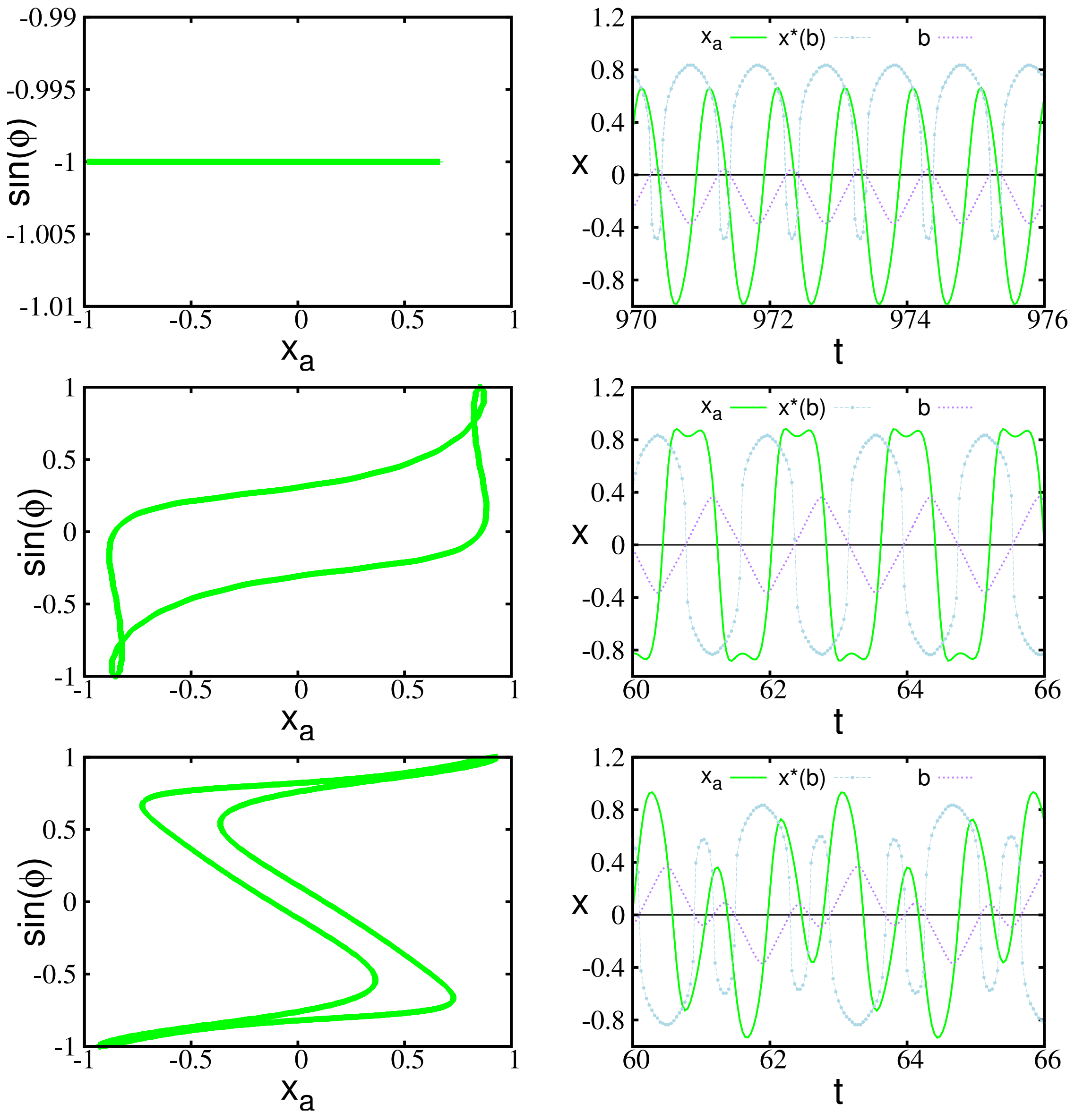


Figure 3.5: Phase diagrams, e.g the sine of the angle ϕ between the ground and the internal axis (see Fig. 2.1) against the actual position x_a of the pendular (left column) and the position x_a of the small pendular as a function of time. (The light-blue dots denote the adiabatic fixpoints $x^*(b)$ of the subsystem (x,v,b) not the output x_t of the controller) against the time t (right column) for $\lambda = -25$ and $\epsilon_b = 0.7$. The first/second/third rows show the zero/1to1/1to3-modes respectively.

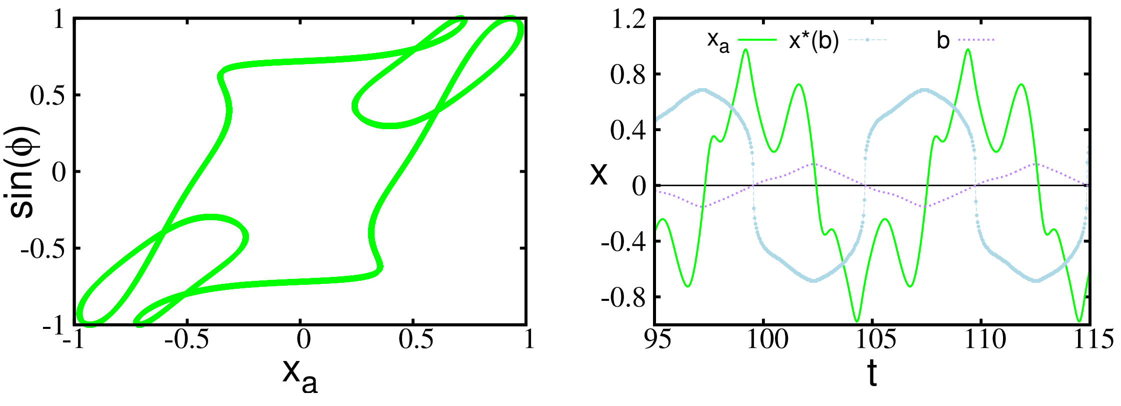


Figure 3.6: The bf-mode for $\epsilon_b = 0.05$. The adiabtaic fixpoints $x^*(b)$ are modulating the dynamics, while the rolling motion and the embodiment of the barrel leading to the smaller oscillations.

3.2.1 Rolling modes

The simulations are done for different values of ϵ_b and λ. In most of the simulations, in the first few seconds a force is applied to the robot manually; leading it to different, stable rolling modes. In all the simulations the robot could be forced into the zero-mode. In this mode the internal axis is perpendicular to the ground, hence the robot is in a resting state while the pendular is oscillating along the axis. The oscillation is slightly shifted from the origin, because gravitation pulls the pendular down. For high values of ϵ_b this is the only accessible mode. There is also a range of ϵ_b values for which two other modes occur, where the barrel is rolling in one direction. According to the ratio of the number of oscillations of the pendular and the rolling oscillations of the whole robot these modes are refered to as one-to-one-mode (1to1-mode) and one-to-three-mode (1to3-mode).

Three different modes have been recovered for parameters $\lambda = -25$, $\epsilon_b = 0.7$ and $a = 2$ (see Fig. 3.5). The plots of the measured probabililty distributions are noted in the appendix (Fig. A.2), they are not converging to the Gaussian target distribution. In the case of the 1to1-mode one oscillation of the pendular corresponds to one turn of the barrel and respectively in the case of the 1to3-mode three oscillations of the pendular correspond to one turn of the barrel.

In the zero-mode the barrel is not rolling and it is similar to the mode examined for the standing barrel, the only difference being the shift in the centre of mass due to gravitaional force. The other two modes, especially the 1to3-mode, only occur together with the rolling motion. For small ϵ_b the barrel can achieve another mode, where it oscillates by rolling around a centre point, while not converging to the resting zero-mode (see Fig. 3.6). This mode is denoted by bf-mode (back-and-forth-mode).

3.2.2 Speed of the modes

An important question is, in which way can one drive the barrel to some specific state by controlling the adaption rates carefully. Here it is examined how the average speed of the barrel depends on ϵ_b.

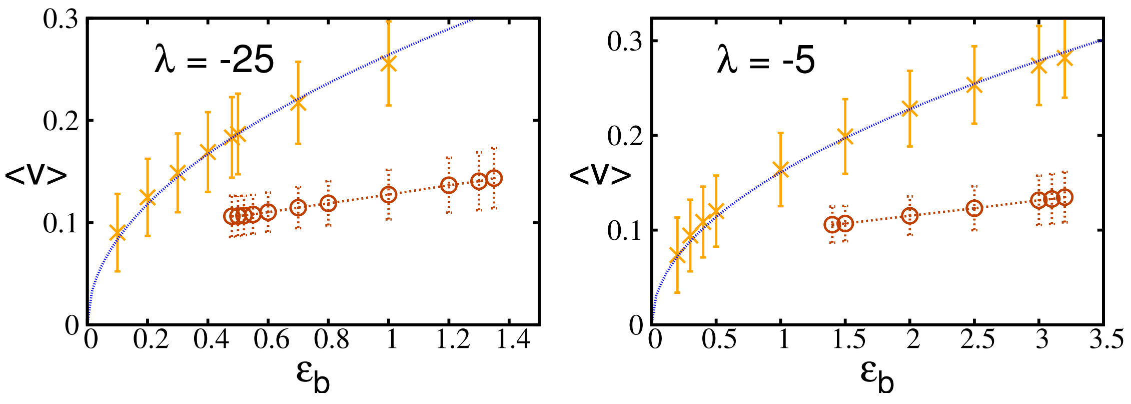


Figure 3.7: The average speed of the rolling modes is plotted as a function of the ϵ_b adaption rate. Crosses and circles correspond to the measured values of the 1to1-mode and the 1to3-mode respectively. The blue curve, as defined by $c\sqrt{\epsilon_b}$ with $c = 0.2642$ on the left and $c = 0.161$ on the right, seems to approximate the measured values of the 1t1-mode nicely. *Left*: Plots for $\lambda = -25$, compare to Fig. 3.5. The 1to1-mode is stable for $\epsilon_b \in [0.1, 1]$ while the 1to3-mode for $\epsilon_b \in [0.5, 1.35]$. For $\epsilon_b < 0.1$ the bf-mode occurs. The zero-mode occurs for all values (for $\epsilon_b > 1.35$ it is the only observed mode). *Right*: Plots for $\lambda = -5$. The 1to1-mode is stable for $\epsilon_b \in [0.2, 3.2]$ and the 1to3-mode only for $\epsilon_b \in [1.4, 3.2]$. For $\epsilon_b < 0.2$ the bf-mode occurs. The zero-mode occurs for all values (for $\epsilon_b > 3.2$ that is the only observed mode).

The average speed of the zero-mode is ostensibly 0. The 1to1-mode has the highest average speed, the pendular is always shifted in the rolling direction, while in the case of the 1to3-mode the pendular is sometimes shifted in the opposite direction and hence decelerates the barrel. For small values of adaptation, $\epsilon_b < 0.05$, the rolling modes disappear and are replaced by the bf-mode. For high values of adaptation, $\epsilon_b > 3.3$, only the zero-mode is stable. Consequently, it is difficult to derive an analytic relation between the average speed and the adaptation rate. Still, numerical simulations suggest a scaling rule

$$v \sim \sqrt{\epsilon_b} \tag{3.13}$$

for the average speed of the barrel in the 1to1-mode (see Fig. 3.7).

3.3 Barrel with two internal axes

As a next step the complexity of the system is increased. A second internal axis, also with a small pendular on it, is added orthogonal to the first one (see Fig. 3.8), with a slider servo to move the second pendular. The robot is able to move only along the global x-axis, but now there are more options to change the barycenter of the robot. Two identical and independent neurons with the learning rules discussed in Sec. 2.18 control each of the motors.

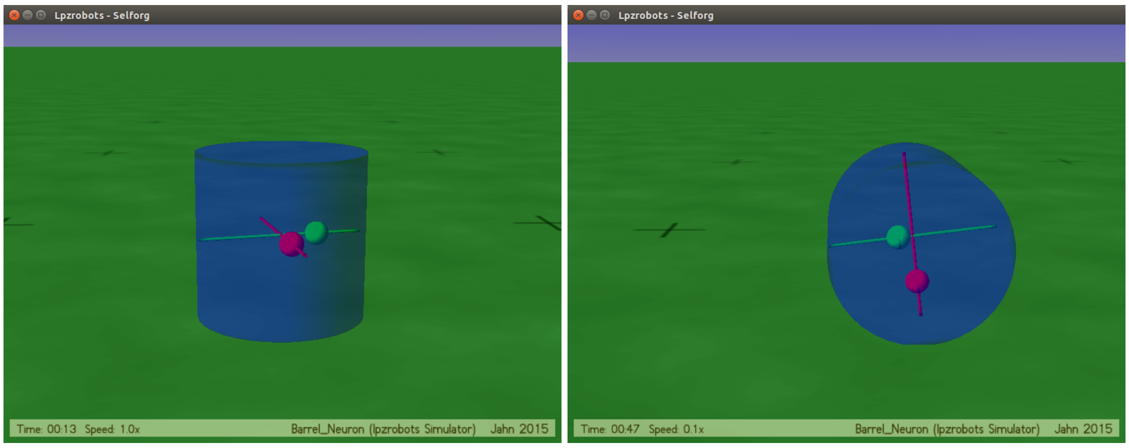

Lpzrobots - Selforg
Time: 00:13 Speed: 1.0x
Barrel_Neuron (lpzrobots Simulator) Jahn 2015
Lpzrobots - Selforg
Time: 00:47 Speed: 0.1x
Barrel_Neuron (lpzrobots Simulator) Jahn 2015

Figure 3.8: Barrel2Masses from LPZROBOTS, the motors that move the pendulars are controlled independently by one neuron each. *Left*: The barrel standing on top, thus the influence of the environment is minimised. *Right*: The barrel in rolling motion. Note the phase-shift between the two pendulars.

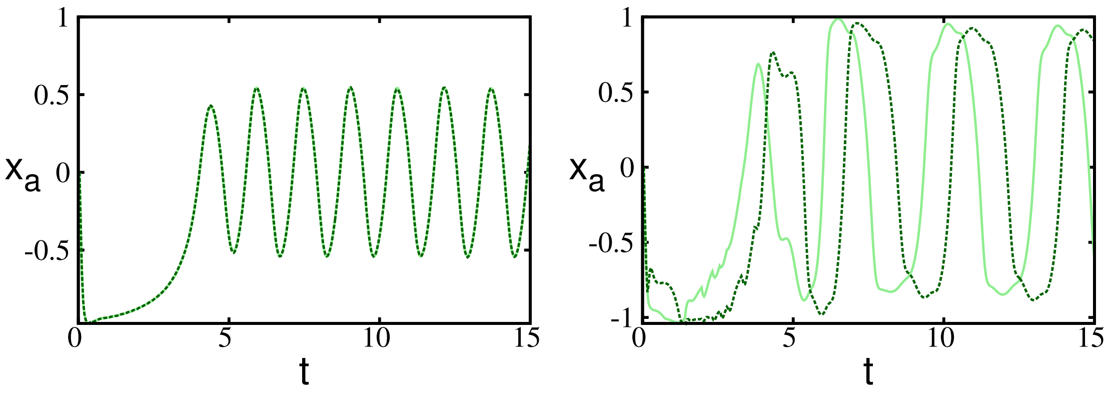


Figure 3.9: The controller is initialised with exactly the same parameters ($\lambda = -5$, $\epsilon_b = 0.5$) for both neurons. The light-green solid and the dark-green dotted curves denote the actual positions of the pendulars repectively. *Left*: The robot is placed on top. The two pendulars fullfill an identical motion (both curves fall on each other). No self-organisation effects occur due to the lack of coupling. *Right*: The barrel is placed on its side. As an effect of the feedback of the robot and the environment, a constant phase-shift emerges and the barrel peforms a rolling motion (compare Fig. 3.8).

While the neurons do not get any direct information from each other they are implicitly coupled through the robot and the environment. The state of one neuron changes the state of the robot or environment, which then has an effect on the second neuron and vice versa. This can be compared to the swarming of fish in biology. Each fish is an independently acting agent, nonetheless the swarm motion of fish is higly structured. The question is wether in the case of the barrel being controlled by two neurons, do self-organised effects occur?

To examine this the barrel is first placed on top (see left plot of Fig.3.8). In this position the feedback of the environment is minimised, since the motion of the pendulars has nearly no effect on the state of the robot (see Fig. 3.8 left plot). The two motors act independently. When the barrel is placed on its side (see Fig. 3.8 right plot), moving the pendulars has a strong effect on the motion of the robot in the environment, hence one can expect some self-organisation. Fig. 3.9 shows the position of the pendulars with respect to time. In the first case the neurons behave identically. This is unsurprising, since they are both started with the same initial conditions. In the second case as an effect of the environment, the two neurons perform the same motion, but now with a small phase-shift, while the barrel is in a smooth rolling motion.

Finally ϵ_b is varied (see Fig. 3.10). First a value is chosen, such that the barrel is in a smooth rolling motion ($\epsilon_b = 0.25$ upper right plot). Increasing ϵ_b leads to faster oscillations of the pendulars and a higher average speed of the barrel ($\epsilon_b = 4$ upper right plot). At some point the rolling motion becomes unstable, the barrel performs small 'jumping' motions with a very low average speed ($\epsilon_b = 8$ lower left plot). Here again an effect of mismatch of timescales occurs. But after decreasing ϵ_b the barrel quickly joins the rolling motion again ($\epsilon_b = 0.5$ lower right plot).

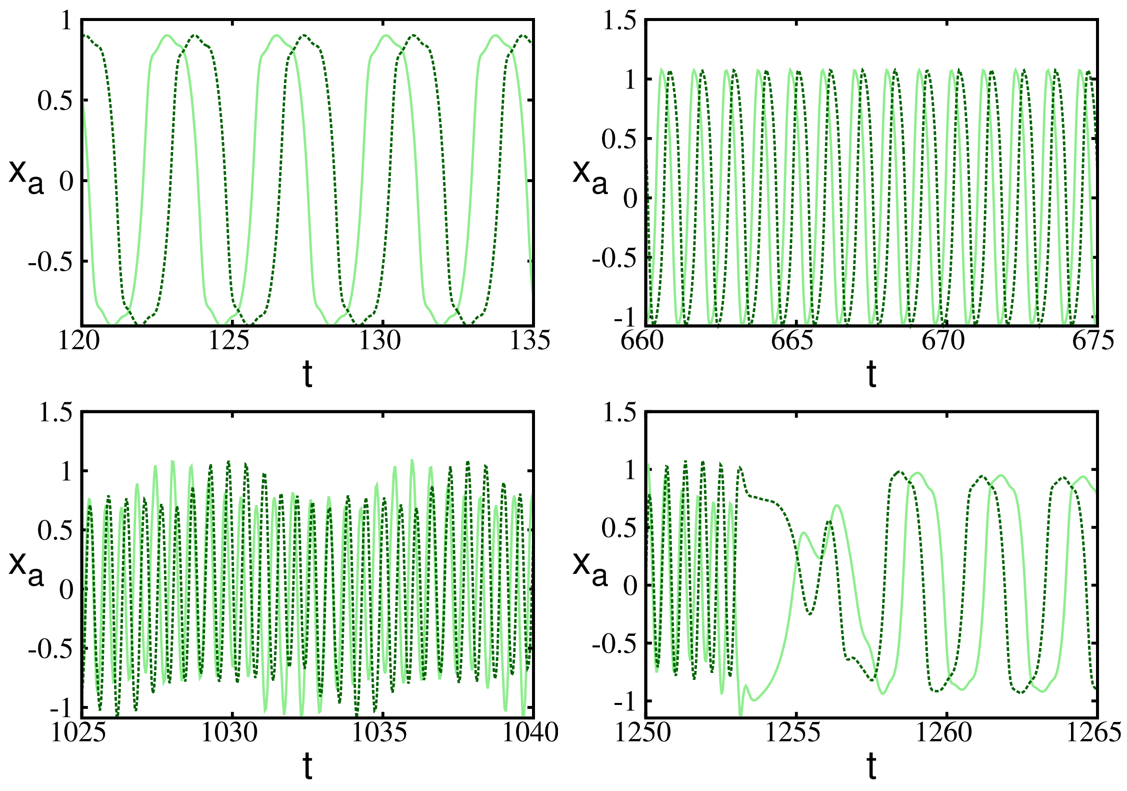


Figure 3.10: The barrel with two axes in rolling motion (*Upper left*: Slow motion ($\epsilon_b = 0.25$, $< v >= 0.0952$. *Upper right*: Fast motion ($\epsilon_b = 4$, $< v >= 0.3261$)). *Lower left*: The fast rolling motion becomes unstable ($\epsilon_b = 8$, $< v >= 0.0457$), another slow but unsmooth rolling mode emerges. *Lower right*: ϵ_b is decreased from 8 to 0.5 at $t = 1252$. The pendulars return quickly to the constant phase-shift.

4 Summary and Discussion

One goal of this work was to launch the LPZROBOTS simulator in Frankfurt and to prepare the foundations for further investigations. It turned out that this software can be very useful to test some parts of the theory developed previously in this group.
An important thing is, that the simulations are realistic and theoretically they can be repeated with real robots. More specifically, the aim of this thesis was to model an elementary sensory-motor loop embedded to the LPZROBOTS environment using simple robots controlled by adapting neurons. The model can be sketched as follows.
The brain of the robot (the neuron) gets some input x_a from the environment and using this, generates an output x_t to reach some desired state. The body (the servos), then tries to bring it to the new state (important: this is not happening instantaneously!). The movements are performed using somewhat realistic motions, in our case the pendular is controlled through a spring. This is similar to the way humans move their extremities for example.
We show that already a very simple robot controlled by a single neuron performs a non trivial dynamics. Minimising the influence of the environment, considering only the small pendular on the axis, the dynamics is driven by the rapid shifts of the adiabatic fixpoint under the slowly adapting internal threshold b, which yields to the oscillation movements of the pendular. Taking into account the environment, the neuron drives the robot to a natural motion of the barrel - the rolling mode. It is possible to drive the robot to different modes by controlling just one parameter, the ϵ_b adaptation rate. For small values we got the bf-mode, for intermediate values the rolling modes are dominating and for high values the resting mode was the only possible state. These modes can be compared to different motions of humans, like standing, walking and running. Walking and running are not just distinguishable by the different speed, they are intrinsicelly different dynamic patterns.
We used the learning rules derived by Linkerhand & Gros [LG13], which are based on the principle of entropy maximisation of the adapting firing rates. Since we considered a slightly different system (no exponential decay of the membrane potential), the measured statistics are not converging to the desired target distribution. It would be interesting to test other learning rules, which yield to such a convergence.
At the end of the thesis we present briefly a more complex system with more than one neuron involved. Although there is no direct connection between the two neurons they are coupled strongly through the environment. After starting the barrel from a natural rolling position, the two independent neurons interact immediately and organise themselves in a way such that smooth rolling motions of the barrel occur.
The next step would be to consider larger systems, other learning rules and different motors. One of the current projects of the group deals with a snake-like robot with many ankle segments, each controlled by a neuron, which show strong effects of self-organisation.

Bibliography

[DFT90] Doyle, J. ; Francis, B. ; Tannenbaum, Allen: *Feedback Control Theory.* 1. Macmillan Publishing Co., 1990

[DM11] Der, R. ; Martius, G.: *The Playful Machine.* 1. Springer, 2011

[GLW14] Gros, G. ; Linkerhand, M. ; Walther, V.: Attractor Metadynamics in Adapting Neural Networks. In: *Artificial Neural Networks and Machine Learning - ICANN 2014* (2014)

[Gro10] Gros, C.: *Complex and adaptive dynamical systems: a primer.* 4. Springer, 2010

[LG13] Linkerhand, M. ; Gros, C.: Self-Organized Stochastic Tipping in Slow-Fast Dynamical Systems. In: *Mathematics and Mechanics of Complex Systems* 1 (2013), Nr. 2, S. 129–147

[MG13] Markovic, D. ; Gros, C.: Self-Organized Intrinsic Adaptation in Autonomous Recurrent Neural Networks. In: *Neural Computation* 24 (2013), S. 129–147

[Rü11] Rüger, R.: Monte Carlo Methoden in der statistischen Physik und ihre Anwendung zur Simulation von Spinsystemen. (2011)

[SK99] Stemmler, M. ; Koch, Ch.: How voltage-dependent conductances can adapt to maximize the information encoded by neural firing rate. In: *Nat. Neuroscience* 2 (1999)

[Wal13] Walther, V.: Metadynamics of Attractors with an Application to Neural Networks. (2013)

Erklärung

Nach § 30 (12) der Prüfungsordnung für den Bachelor- und Masterstudiengang Physik der Johann-Wolfgang-Goethe-Universität Frankfurt am Main erkläre ich, dass diese Arbeit von mir selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst wurde. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderen fremden Texten entnommen wurden, sind als solche kenntlich gemacht. Weiter erkläre ich, dass die Arbeit nicht – auch nicht auszugsweise – für eine andere Prüfung verwendet worden ist.

Frankfurt am Main, den 25. Mai 2015

(Tim Jahn)

Appendix

A.1 Parameters

A.1.1 Robot

The robot class is inherited from sphererobot3masses.

```
c.diameter     = 1;
c.spheremass   = 1;   //mass of the body of the robot (the cylinder)
c.pendularmass  = 1.0;  //mass of the servo pendular (small sphere)
c.pendularrange  = 0.3; // range of the slider from center
                        //  in multiple of diameter [-range,range]
c.motorpowerfactor = 200;
c.motorsensor = true;
c.brake=0;
c.axesShift=0;
```

A.1.1.1 Joint

```
dParamLoStop     = -conf.diameter*conf.pendularrange
dParamHiStop     = conf.diameter*conf.pendularrange
dParamStopCFM    = 0.1
dParamStopERP    = 0.9
dParamCFM        = 0.001
```

A.1.1.2 Servo

```
_min          = -conf.diameter*conf.pendularrange
_max          = conf.diameter*conf.pendularrange
KP            = conf.pendularmass*conf.motorpowerfactor = 200
KD            = 0.1  //(this is slightly subcritical, critical
                     // damping for KD = 2 sqrt(spheremass/KP) = 0.14142135
KI            = 0
```

[-pendularrange,pendularrange] is the region where the servo can move the centre of the spring (the domain of the targetposition). [dParamLoStop,dParamHiStop] is the region where the small pendular, which is fixed to the spring, can move. In our case we chose these regions to be equal.

A.1.2 Controller

```
a           = 2
epsilon_a    = 0
epsilon_b \element (0,4)
lambda    \element {-5,-25}
```

A.1.3 Simulation

```
cameraspeed          = 100.000000  camera speed
fps                  = 25.000000  frames per second
friction             = 0.100000  rolling friction coefficient
gravity              =-9.810000  strengh of gravity (-9.81 is earth gravity)
noise                = 0.000000  global noise strength
randomseed           = 1428564641  random number seed (cmdline -r) (readonly)
realtimefactor       = 1.000000  speed of simulation wrt. real time (0: full speed)
simstepsize          = 0.010000  stepsize of the physical simulation (in seconds)
controlinterval      = 4         interval in steps between subsequent controller calls
```

A.2 Manual

Here we provide a step by step installation guide for the LPZROBOTS package. Also check the webpage http://robot.informatik.uni-leipzig.de and http://robot.informatik.uni-leipzig.de/software/doc/html/index.html for further informations.

A.2.1 Installation

Go to https://github.com/georgmartius/lpzrobots and copy the link in the white box on the right side (something like https://github.com/georgmartius/lpzrobots.git). Open a terminal and type

```
git clone https://github.com/georgmartius/lpzrobots.git
```

It will start downloading the newest version in a folder called ‘lpzrobots’. Enter the folder and type

```
make all
```

it will ask where to install, choose

```
/usr or /usr/local
```

if you have sudo rights, else install it to your home directory

```
/home/yourlogin
```

In this case you have to modify your .bashrc file. Go to your home directory and type

```
vim .bashrc
```

scroll to the end and insert

```
export PATH=/home/yourlogin/bin:$PATH
export LD_LIBRARY_PATH=/home/yourlogin/lib
```

and restart the process. For the installation mode chose ‘d’ and confirm. It will start the installation (it can take over one hour). When it is finished go to a simulation folder and type

```
make
```

It will create an executable file called ’start’. You can run the simulation with

```
./start
```

Be careful, many simulations are not updated and hence do not work. /lpzrobots/ode_-robots/examples/basic and /lpzrobots/ode_robots/simulations/barrel should work for example. Check the tutorial for the example called ‘basic’ (http://robot.informatik.uni-leipzig.de/wiki/doku.php?id=tutorials).

A.2.2 Creating a Simulation

Go to /lpzrobots/ode_robots/simulation. It makes sense to copy an existing template simulation first (for example template_sphererobot). Call

```
./createNewSimulation.sh template_sphererobot mysimulation
```

here ‘mysimulation’ is the name of your simulation. This will copy all files which are necessary. Do not change anything in the libraries. If you want to change something, copy the file into your simulation directory and rename it (of course you have to rename the class too). You have to adapt the includes (if the included file is in the same directory

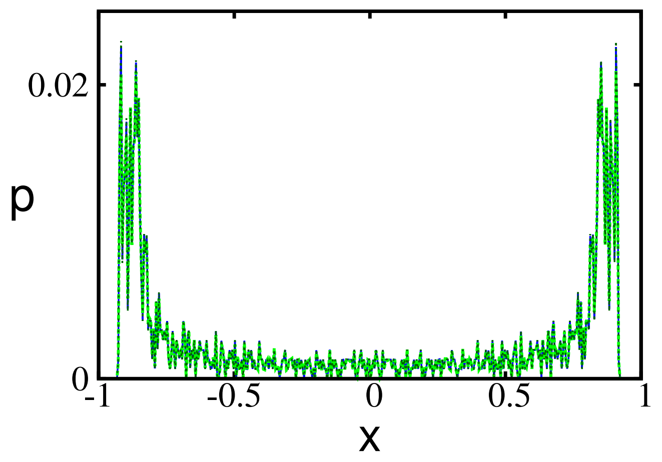


Figure A.1: The measured probability distributions of the 1t1-mode for the whole length of the simulation (blue curve) compared to the data corresponding to before (light-green curve) and after (dark-green curve) turning for $\lambda = -25$ and $\epsilon_b = 3.15$. The curves are falling on each other. For the statistics 43500 (90000 for the whole length of the simulation) data points are binned (200 bins).

use #include "example.h", if it is from a library use #include <ode_robots/example.h> or #include <selforg/example.h> respectively). You have to add the files you copied to your directory (which you want to compile) to Makefile.conf .

```
FILES = main example1 example2
```

A.3 Unexplainable Behaviour

Theoretically, the barrel examined in this thesis should just be able to move along the x-axis. Still, for bigger values of ϵ_b and λ, while rolling, the barrel suddenly turns around a small angle ($\sim \pm 30°$). This happens at different timepoints for the same parameters (but with different initial conditions, since the barrel is forced into the rolling motion manually). We have not found any explanation for this behaviour. At least the turning does not change something in the running simulation. Fig. A.1 shows the measured statistics for a simulation in which the barrel has turned.

A.4 Measured Probability Distributions

Here the measured probability distributions of the modes are presented.

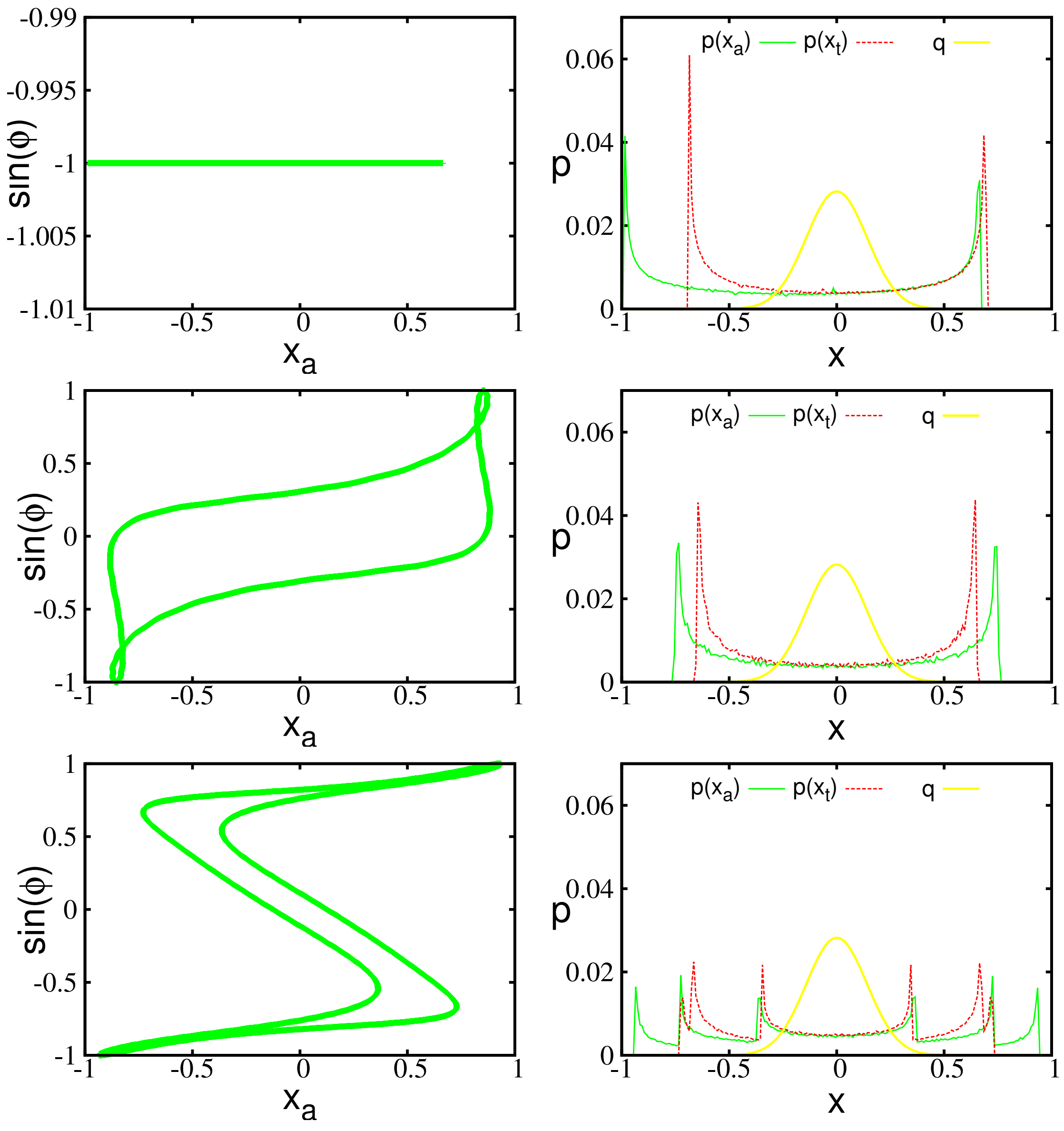


Figure A.2: Phase diagrams as in fig. 3.5 (left column) and measured probability distribution (right column). The red and green curves denote the distributions of the target $p(x_t)$ and actual $p(x_a)$ positions respectively. Both are different from the target Gaussian distribution q (yellow). Here it becomes clear how the dynamic works. The pendular moves fast on the straight lines and slowly at the turning points. For the statistics 43500 data points are binned (200 bins).

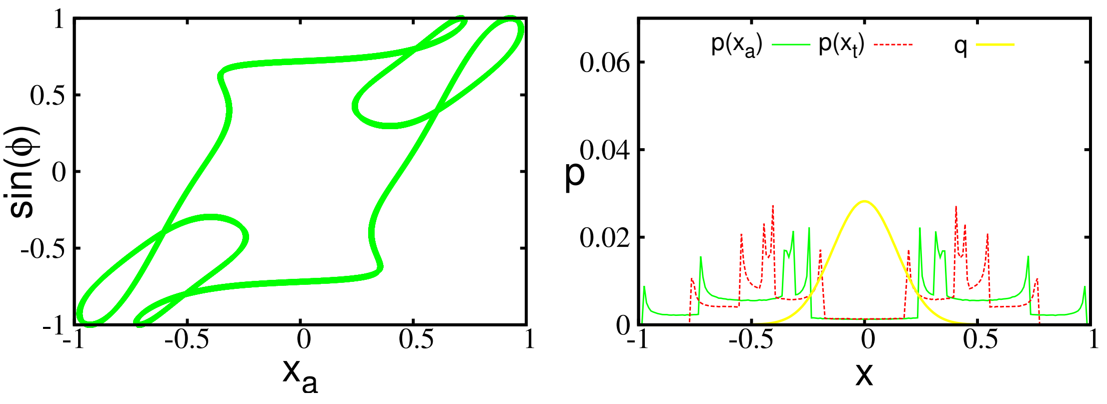


Figure A.3: The bf-mode for a low adaptation rate $\epsilon_b = 0.05$. For the statistics 43500 data points are binned (200 bins).