

Chapter 3

The Sensorimotor Loop

Abstract: This chapter aims at providing a basic understanding of the sensorimotor loop as a feedback system. First we will give some insights into the richness of behavior resulting from simple closed loop control structures in a robotic system called the BARREL. This richness is a lesson we can learn from dynamical systems theory: even very simple systems can produce highly complicated behavior. Nearly everything is possible in such a feedback system that is provided with enough energy from outside. Surprisingly, this is accomplished even with extremely simple, fixed controllers, to which we will restrict ourselves here. In later chapters we will see how the homeokinetic principle makes these systems adaptive and drives them towards specific working regimes of moderate complexity, loosely speaking somewhere between order and chaos.

The aim of this chapter is to make the reader familiar with the general structure and specific properties of tightly coupled sensorimotor loops. In these loops the motor commands are directly related to the sensor readings, so that the robot with its “brain” forms a feedback system. In this context the framework of dynamical systems, known from mathematics and physics, started to get increasing attention in the last two decades [13, 81, 120, 152]. It is a powerful method to analyze [75] and construct [39, 72, 78, 81, 152, 160] robot controllers, as it allows one to formulate the time evolution of the system, in a quantitative manner and to obtain both analytical and qualitative predictions. Dynamical system theory also led to the application of chaos control and coupled chaotic oscillators to robotics [91, 138, 159].

After a general introduction of closed loop control, using the framework of dynamical systems, we study a specific example, the BARREL, that is particularly interesting by its strong embodiment effects. The general settings are investigated subsequently in an elementary sensorimotor loop, a one-dimensional system controlled by a single neuron. Without noise, a pitchfork bifurcation occurs and a pronounced hysteresis behavior is established if the neuron has a bias. We introduce our concept of an effective bifurcation point to allow for noise effects. This concept defines a specific working regime, where robots can already take decisions while still being sensitive to perturbations by the environment. Eventually, we present briefly neural networks as universal tools for the realization of the control system. The investigations are based on dynamical systems theory but the mathematics will be kept

simple and essentially self-contained so that no special knowledge of the latter is necessary.

The theoretical studies are underpinned by robotic experiments that can be executed with our simulation environment. Experiments are provided for studying closed loop control and in particular the concept of the effective bifurcation point under various conditions. The role of the embodiment can be investigated with wheeled robots. When interconnected to form a chain of robots emergent cooperativity is observed even though decentralized control is used. In this way, the present chapter makes first contact to the central idea of externalizing complexity, namely to control complex physical modes with very simple control structures and thus to source the complexity out to the interaction with the environment.

3.1 Sensorimotor Loop — The General Case

In a self-consistent approach to self-organizing robot behavior, the sensor values are the only source of information for the robot. This is also the *credo* of our approach. The communication between the “brain” and the body of the robot takes place at the discrete instants of time $t = 0, 1, 2, \dots$. In practice, typical clock frequencies are ranging from 10 to 100 Hz, depending on the speed of the information processing. In each time step a vector of sensor values $x_t \in \mathbb{R}^n$ is reported. Let us illustrate this by two examples. Firstly we consider a wheeled robot with sensor vector

$$x = (v_l, v_r, s_1, \dots, s_k)^\top \quad (3.1)$$

where v_l and v_r are the wheel velocities of the left and right wheel, respectively, as measured by the wheel counters, and s_i are the values of the infrared sensor i with $0 \leq s_i \leq 1$. The wheel velocities are examples for proprioceptive sensors. Infrared sensors are examples of exteroceptive sensors since they get information about the relation to the outside world. In many of the applications treated further below the robot has only proprioceptive sensors providing informative feedback on the state of its body, an example being our dog robot, see Fig. 3.1, where

$$x = (x_1, \dots, x_n)^\top, \quad (3.2)$$

x_i are the joint angles.

3.1.1 The Controller

At each time step t , the controller sends a vector $y_t \in \mathbb{R}^m$ of target values to the motors of the robot. Closed loop control means that the controller is given by a function $K : \mathbb{R}^n \rightarrow \mathbb{R}^m$ mapping sensor values $x \in \mathbb{R}^n$ to motor values $y \in \mathbb{R}^m$. In the most simple case this is a function

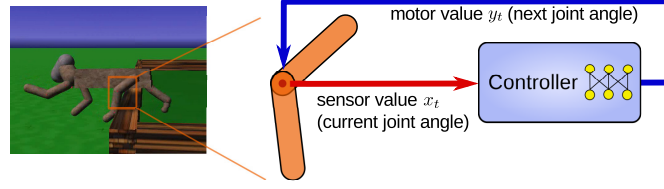


Fig. 3.1: Example of a sensorimotor loop with the joint angles as proprioceptive sensors. Only a single joint is depicted.

$$y = K(x) \quad (3.3)$$

depending in general on a vector of parameters p that can be adapted in order to realize a desired behavior. More generally, the map may depend on an internal state $s_t \in \mathbb{R}^k$, which is updated in each time step as well so that the controller is realized as

$$y_t = K(x_t, s_t) \quad (3.4)$$

$$s_t = O(x_t, s_{t-1}) \quad (3.5)$$

where now $K : \mathbb{R}^n \times \mathbb{R}^k \rightarrow \mathbb{R}^m$. Controllers with internal states are realized conveniently by recurrent neural networks as introduced later in Sect. 3.6.4. However, the main problem in using an internal state is to find the update rule (3.5) for the latter such that the system develops the desired behaviors.

We will use a very simple realization of the parameterized controller function $K(x)$, see Eq. (3.3), but complement it later with a dynamics for the parameters p , which will be a function of the state dynamics. We are free to consider the parameters p as internal state variables so that the new feature introduced by the homeokinetic principle is an explicit rule for the function $O : \mathbb{R}^n \times \mathbb{R}^k \rightarrow \mathbb{R}^k$ in Eq. (3.5).

3.1.2 Forward Model and Sensorimotor Dynamics

Let us stipulate that our robot has a certain ability for cognition. We understand here cognition in a minimalist sense as the ability of the robot to predict the consequences of its actions in the near future with a forward model. Formally, this is realized by a function $M : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ mapping the sensor values x and the actions y of the robot onto the new sensor values, i.e.

$$x_{t+1} = M(x_t, y_t) + \xi_{t+1} \quad (3.6)$$

where ξ_t is the difference between the predicted and the true sensor values. This quantity will also be called noise because it contains the unpredictable effects like sensor noise and so on.

With these notions we may write the dynamics of the sensorimotor loop in the closed form

$$x_{t+1} = \psi(x_t) + \xi_{t+1} \quad (3.7)$$

where

$$\psi(x) = M(x, K(x)) . \quad (3.8)$$

The function $\psi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is called the dynamics model and can be understood as a time series predictor for the time series of the sensor values x_t , see also Fig. 3.2.

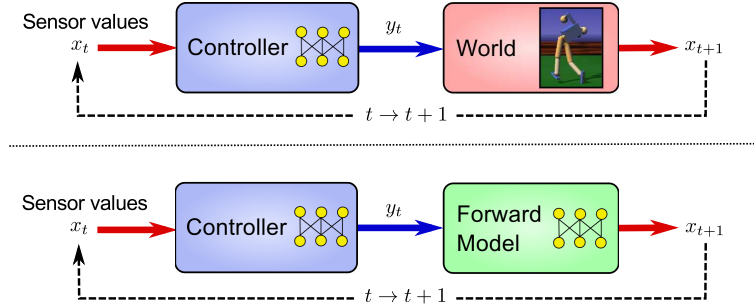


Fig. 3.2: The sensorimotor loop (top) and its model (bottom).

The prediction error, also called the model error, can be defined as (dropping the time indices)

$$E_{\text{pred}} = \xi^\top \xi . \quad (3.9)$$

Let the model M be realized by a parameterized function with parameters $a \in \mathbb{R}^M$. In order to minimize the prediction error (3.9) the parameters can be adapted by following the gradient of the error function in descending direction, see Fig. 3.3 for the signal flow. Let a be any of those parameters, then the learning step is defined as

$$\Delta a = -\varepsilon_A \frac{\partial E_{\text{pred}}}{\partial a} , \quad (3.10)$$

with a learning rate ε_A that is kept so large that fast parameter changes are possible in the following applications. The model and the learning dynamics can be realized for instance by a neural network as introduced in Sect. 3.6.

The structure of the model and the learning procedure define the passive cognitive abilities of the robot as will be worked out in more detail in the following.

We use the word *model* in two different contexts. There is the *forward model* M that predicts the outcome of the actions. For simplicity it is sometimes called

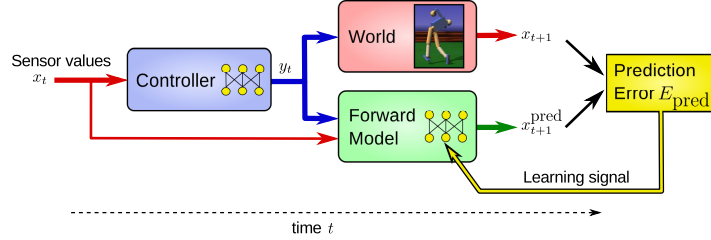


Fig. 3.3: The sensorimotor loop and the forward model of the robot. The controller receives the current sensor values and generates corresponding motor values, which are sent both to the robot and its forward model. The difference between the predicted new sensor values and the measured ones forms the prediction error E_{pred} . A learning signal for the improvement of the model is derived by gradient descending the error E_{pred} .

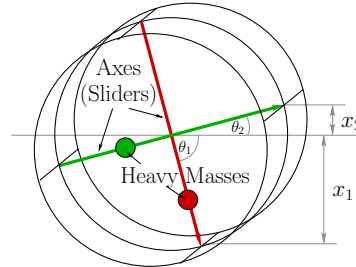
just the model. In its most simple realization it only receives the motor values (actions), in which case it can be considered as a self-model, especially if the sensors are essentially only the proprioceptive ones. The second context is the model of the entire sensorimotor dynamics $\psi(x)$, which we call the *dynamics model*. Quite generally, we will call the robot together with its controller and the forward model the *brain-body system*.

In order to illustrate the formalism of the sensorimotor loop and the merits of closed loop control we consider now an example using a robot with strong embodiment effects, i. e. a robot where the physical effects like inertia and centrifugal forces are heavily interfering with the effects of the actions taken. Afterwards we will return to a simpler case for analytical considerations.

3.2 Dominated by Embodiment: The BARREL

While our emphasis lies on controlling more complex robots we will in the current chapter restrict ourselves to a case of moderate complexity so that analytical considerations are still possible. The machine we are going to use is the BARREL (which is short for barrel robot), see Fig. 3.4.

Fig. 3.4: The BARREL. It consists of a cylindrical encasement with two weights sliding on two orthogonal axes perpendicular to the cylinder axis. Each of the weights is moved by a linear motor so that the system can shift its center of gravity. The sensor values measure only the inclination of the axes, i. e. $x_i = \sin(\theta_i)$. Thus, the true physical state of the system is largely unknown to the controller. Note that $x_1 < 0$ in the displayed situation.



The robot has a cylindrically shaped body. Inside there are two weights that are moved along the two axes by simulated linear motors. Each bare motor is supported in doing its task by an extra PID controller, see Sect. 16.4.4 (p. 301) for details, that compensates for both overshooting and undershooting in the movement of the weights. Nevertheless, due to a limited maximal force the motors cannot move the weights with arbitrary velocity and precision. The movements of the weights induce a change of the center of gravity that causes the robot to roll in one direction or the other.

The BARREL can unfold many different kinds of motion, despite its simple construction, as will be seen below.

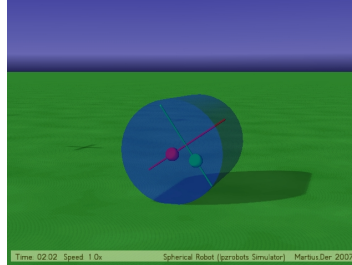
3.2.1 Properties

The reason why we have chosen the BARREL is because it is a body-dominated system. What we mean by this is that, similar to many systems used in robotics, the consequences of an action are largely dominated by the current physical state of the robot. By way of example the effect of shifting a weight on its axis will be very different if the system is at rest or in a rapidly rolling motion so that the sensor reaction on any action is highly ambiguous. The situation is even more complicated due to the physical properties of the robot which are simulated realistically by the ODE physics engine [154] embedded in our simulation software, see Chap. 16. The execution of an action (moving the weights) is largely influenced by the inertia of the weights, the Coriolis forces due to the motion of the weights on a rotating axis, centrifugal forces if the BARREL is rolling with high velocity, and others.

In the following we are going to discuss different control paradigms using the example of the BARREL. This allows us on the one hand to demonstrate some features of the embodied artificial intelligence approach in a simple and transparent manner and on the other hand to outline the perspectives of self-organization for extending this approach to a wider field of applications.

3.2.2 Open Loop Control

One way to control the BARREL is in the open loop setup where a sequence of motor signals is sent to the robot. This sequence may be for instance generated by a central pattern generator. Let us first assume that the BARREL is to roll with a fixed velocity. This may be achieved by shifting the internal weights periodically with a convenient frequency and a phase shift of $\pi/2$. The velocity of the BARREL is then determined by that frequency – one rotation of the BARREL corresponds to one period of the pendular oscillations.



Video 3.1: Open loop control of the BARREL The robot is controlled by a periodic control signal driving the internal weights with a phase shift of $\pi/2$. Starting with a very low frequency of the controller signal, the frequency is doubled at time 2:15, 2:40, 3:05, and 3:30. At 3:45 the BARREL is accelerated by a force (red dot) but is seen to return rapidly to the original mode of behavior. The higher frequencies very clearly reveal the difficulties in realizing a fast motion under the open loop control paradigm. The video can be watched at <http://playfulmachines.com>.

When doing so we find that, at low frequencies and with some friction at least, the BARREL adapts its (average) rotational frequency to the frequency of the pendular oscillations indeed, see Video 3.1.

This behavior is stable against moderate perturbations. To understand this, imagine a stroboscopic mapping depicting the BARREL every moment the red weight, say, has maximal downward elongation. In an ideal and constant rolling mode (without friction) the red weight will be exactly below the cylinder axis since then there is no torque (the green weight is in the center due to the $\pi/2$ phase shift). If the BARREL is externally decelerated, the stroboscopic mapping will show the weight to rotate slowly against the rotational direction of the BARREL (since the maximal elongation is reached before the tip of the axis reaches ground). Hence a torque is exerted counteracting the slowing down. This stabilization mechanism works just as well if the BARREL is being accelerated from outside, as long as within certain bounds.

Let us try to make contact with the embodied AI paradigm [131] at this point. Its aim is to shift the computational load from the controller to the morphology and physical properties of the embodiment. In the above case this is realized due to the self-stabilization effect—a simple periodic signal generates a stable motion pattern in a complex physical object. If the aim was to realize nice harmonic motion one had two options, either to change the wave form of the control signal (to increase the control effort) or to modify the physical properties of the body.

The embodied AI approach takes the latter route. As the experiments show, for a BARREL with fixed physical properties, there are one or more preferred frequencies where the motion is most harmonic. This is also where minimal control efforts¹ are required. On the other hand, in the sense of the embodied AI, given the frequency, one may change the physical parameters like the PID settings, the forces and pen-

¹ Control efforts are understood in terms of the complexity of the required controller that can generate the behavior.

dular ranges, the mass of the cylinder and so forth in order to get the desired nice harmonic motion.

However, this only works as long as the frequency is not too high. At higher frequencies, the self-stabilization effect is lost and different modes of behavior are induced by the periodic signal. It is here that the physical effects, like the Coriolis force, start to dominate so that the BARREL is driven into irregular behaviors, see Video 3.1. Following the embodied AI paradigm, one would try to modify the physical properties appropriately to make the system obey the simple periodic control signal but there is no simple and/or straightforward solution to that problem.

The experiment can also be performed by the reader as described in Experiment 3.1.

Experiment 3.1: Open loop control of BARREL

Many of the experiments described in this book can be performed by the reader using our simulation software that is online at [103]. Here we will describe the handling of the software in more detail. Choose one of the options described at the website to get the software on your computer. You will get a folder with an `index.html`, which when opened in a browser shows all experiment descriptions and provides links to start them. For each simulation a terminal window and a graphical window opens. The latter shows the rendered scene, see Video 3.1. At the same time the terminal window shows a welcome text and the parameters that are used. While working with the simulator both windows are important. The terminal window allows to check and change parameters via a text-based console, which can be entered by pressing `<Ctrl>+C` in the terminal window. A prompt appears (`>`) and you can type

```
>help<Enter>
```

to obtain a list of possible commands, see Fig. 3.5(b). The graphical window allows to observe and possibly interact with the robots. For a list of keystrokes and mouse actions type `h` (make sure the focus is on this window). Our first experiment is **Open loop control of BARREL**. The simulation is now running with the default parameters: `period=300` given in control steps (1/50 s) and `phaseshift=1` given in multiples of π . Changing parameters of the robot or controller is done by using the pattern "Parameter=Value" on the command prompt. For instance, in order to decrease the period duration (increase frequency) of the sine signal type (after pressing `<Ctrl>+C` in the terminal window)

```
>period=200<Enter>
```

If done correctly, the actually set value is echoed, i.e. `period = 200.0000`, otherwise the parameter name was probably misspelled. Hint: you can use the `<Tab>` key to do automatic completion. In the default situation there is no rolling friction. The barrel does not move with a constant speed but oscillates instead. Switch on the friction by

```
>friction=0.1
```

Now, decrease the period further, try: `>period=100,50,10`. Note, that the robot cannot follow the periodic commands if too fast. You can also change to another control mode for instance by using a colored noise with the parameters

```
>strength=Strength
>color=Correlation time of the noise in 1/50 s
```

Try `>strength=1, >color=100` and disable the sine generator with

```
>amplitude=0
```

Most of the parameters of the simulation can be monitored by starting the GUILOGGER by pressing `<Ctrl>+G` in the graphical window. Then a new window appears where you may tick the boxes for the on-line display of sensor and/or motor values, see Fig. 3.5(a). Alternatively you can invoke the MATRIXVIZ with `<Ctrl>+M`, which is especially handy in highdimensional systems, see Fig. 3.5(c).

3.2.3 Closed Loop Control

Outside the self-stabilization region of a rolling mode, the BARREL does not obey the periodic motor signals any longer and realizes more or less chaotic behaviors.

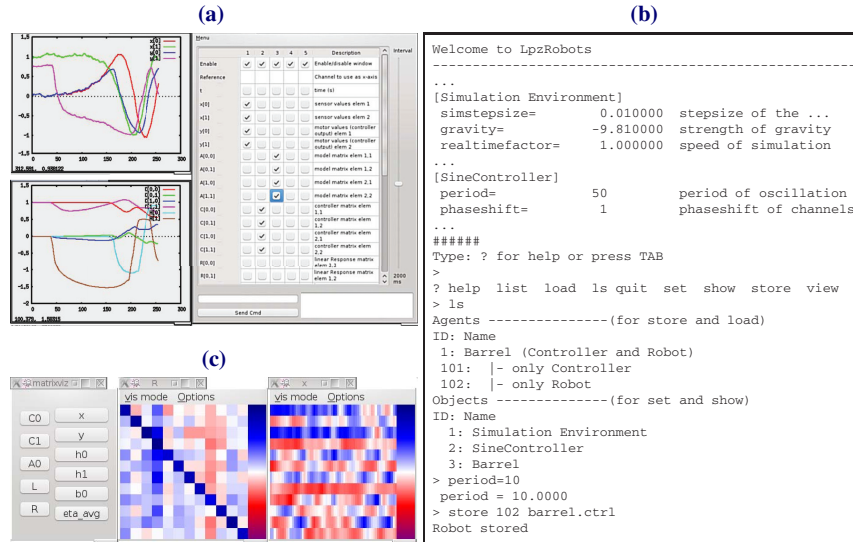


Fig. 3.5: User interface of the LPZROBOTS simulator. (a) GUILOGGER window with two controlled plotting windows. In the main window (**right**) a set of channels are selected. Their temporal evolution is shown in the subwindows (**left**), here sensor and motor values, and the parameters of the controller. (b) Terminal window with console interface. (c) MATRIXVIZ showing the state of the matrix R and the time evolution of x .

In some sense the body takes over the command and if the controller is to achieve a certain objective, it has to “watch” carefully what the body is doing and develop the right “tact” for it. However, this can not be done in the open loop control mode we used in the previous section, because the sensors have to be taken into account. In the special case of the BARREL, we have two sensors measuring the inclination of the axes, see Fig. 3.4.

As before, we assume that the sensor values at time t are comprised in the vector $x_t \in \mathbb{R}^n$. In the case of the BARREL we have $n = 2$ and we may normalize the sensor values so that $-1 \leq x_{ti} \leq 1$ for $i = 1, 2$. If the robot is rolling with a fixed velocity, the vector of sensor values x_t is rotated in each time step by a fixed angle α so that we have the very simple sensor dynamics

$$x_{t+1} = U(\alpha)x_t, \quad (3.11)$$

where U is the rotation matrix

$$U(\alpha) = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$$

rotating a vector by the angle α .

Under the closed loop control paradigm, the controller is a function $K : \mathbb{R}^n \rightarrow \mathbb{R}^m$ mapping the vector x_t of sensor values to the vector y_t of motor values, see Eq. (3.3). These controllers can be rather complex, in particular they may contain internal states that modify the mapping depending on contexts, see Eqs. (3.4, 3.5). In the BARREL case $y_t \in \mathbb{R}^2$ gives the nominal positions of the internal weights on their respective axes. We will use for the moment a very simple controller that will prove sufficient to produce a rolling motion with fixed velocity. The idea is that in the stable rolling mode, the vector of actions y_t is to be in a fixed phase relation to the vector of the sensor values x_t .

Let us therefore tentatively put the controller as

$$K(x) = Cx \quad (3.12)$$

where

$$C = cU(\phi) \quad (3.13)$$

with c defining the amplitude of the weight shifting. When using this controller we find stable rolling modes as expected. We may push the BARREL or reverse its velocity from outside but after a very short time the robot returns to its stable rolling mode with fixed velocity, to reproduce follow Experiment 3.2.

If C is a rotation matrix one observes stable rolling modes with a frequency defined by ϕ , see Eq. (3.13). However, it is observed that ϕ is in general larger than the true rotation angle of the sensor vector in one time step α (3.11). Instead one may choose ϕ rather large without increasing the velocity considerably. The phase difference is due to the specific physical properties of the robot described above and could be evaluated empirically or calculated by knowing the physics of the system in detail. However, this is not in the spirit of this book, which is devoted to the self-organization of specific modes in a physical system without knowing it in advance.

The linear controller, see Eq. (3.12), is appropriate in the BARREL case since the sensor values are not in a direct proportionality to the motor values. In general, we have to make sure the actions (motor values) are kept in bounds by introducing a smooth squashing function $g(u)$, that is applied componentwise putting

$$K(x) = g(Cx + h) \quad (3.14)$$

where $h \in \mathbb{R}^m$ is a bias introduced for greater generality. In this way, motor values are kept safely in a defined interval.

The nonlinearity in Eq. (3.14) does not qualitatively change the behavior if C is chosen as in Eq. (3.13). However, we are now free to choose the matrix elements of C arbitrarily. As the experiments show, depending on the parameters, one observes a variety of behavioral modes, which are sometimes surprising given the extremely simple nature of the controller. An example is the “lolloping” mode, see Video 3.2. The variety of modes is even larger if the bias values h are also chosen arbitrarily, see Experiment 3.2.

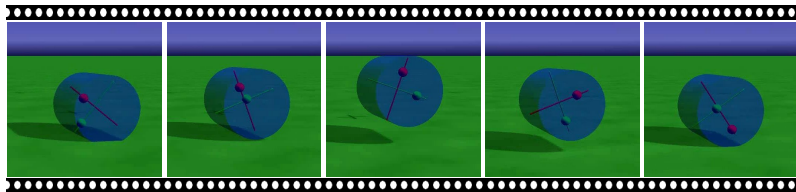
By varying the parameters in a systematic way, we obviously can find out about the scope of modes that are realizable by this type of controllers. In the BARREL

Experiment 3.2: Closed loop control of BARREL

Start the simulation **Closed loop control of BARREL**. The simulation is now running with the C matrix as $C_{11} = C_{22} = 1$, $C_{12} = -C_{21} = .1$ and $h_0 = h_1 = 0$. Change the parameters `coupling1` for the diagonal and `coupling2` for the non-diagonal matrix elements of C , e. g.

```
>coupling1=-0.2
```

Use the GUILOGGER (<Ctrl>+G) for watching the sensor values, which are a good indicative for the behavior of the BARREL. In order to select random control parameters in the interval $(-5,5)$ press `r` on the graphical window. The new parameters are printed on the terminal and you can use the GUILOGGER for monitoring all parameters (x,y,C,h) . To randomize also the bias terms h in the interval $(-3,3)$ press `R` (<Shift>+R), which results in even more different behaviors. With `L` (Shift+L) you can set to $C_{11} = C_{12} = 2$, $C_{21} = C_{22} = -1$, which will cause the robot to roll and jump such that we call it the lolloping mode.



Video 3.2: BARREL performing a “lolloping” behavior. A fixed closed loop controller ($C_{11} = C_{12} = 2$, $C_{21} = C_{22} = -1$, $h_{1,2} = 0$) was used. The robot jumps by rapidly moving the internal weight while rolling from left to right. The video can be watched at <http://playfulmachines.com>.

case the space of parameters is 6 dimensional so that the search space is already quite large. However, in the systems to be considered later, the parameter space is of several hundred dimensions so that an unbiased search is impossible. The new perspective self-organization can bring into this paradigm is to find out, by self-exploration, about the whole range of **low-complexity** modes the system is able to support by low-complexity controllers. These modes may then be viewed as the candidates for embodied AI realizations of more complex behavior architectures.

Before going to the more theoretical considerations let us remark a remote similarity of the BARREL to the famous passive walker [111] when driven by some periodic motor forces enabling it to walk on a horizontal plane [32]. This walker works by having its center of gravity a little ahead of the point of support, falling over being avoided by the leg swinging forward to support the body in the right moment. The BARREL has an easier life since it is always protected from falling over by the cylinder encasement. Nevertheless, in both cases there is a self-stabilizing and self-promoting effect due to the specific embodiment so that the amount of control can be kept small.

3.3 Analyzing the Loop

In order to treat essential features of the closed loop control analytically, we consider now an even simpler example, namely a one-dimensional sensorimotor loop. One special application will be the velocity control of wheeled robots. Actually, this

seems a little strange, given that the realization of an externally determined wheel velocity is a standard task of classical robotics with many reliable solutions. However, our goal is different. We want to use the wheel velocity control as a transparent example for the more involved cases to be considered later and moreover we aim to underline the merits and the potential of closed loop control for embodied systems. Furthermore, we want to show how minimalist decentralized control can give rise to self-organization effects as a mere consequence of the specific physical properties of the embodied system. This will be exemplified by a chain of such robots with each wheel being controlled independently so that control is strictly decentralized. We will see that under certain conditions the robots spontaneously cooperate. This takes place even when moving in a maze with only narrow passages between obstacles. Thereby, the investigation of this special system gives us the opportunity to come into contact with basic notions and effects relevant for the self-organization of robot behavior. Moreover we will also argue that this kind of control opens new perspectives for robotic tasks in quite practical settings.

Let us consider a wheel of a robot rotating with velocity v_t . It is monitored by a wheel counter, which outputs a measured velocity x_t so that

$$x_t = A v_t + \kappa_t$$

where κ_t is the measurement (sensor) noise and A is a hardware constant assumed to be known for the moment. The controller of the robot outputs the target velocity y_t for the current time step. Assuming that the motor is able to realize in one time step the target wheel velocity we would have $v_{t+1} = y_t$ and $x_{t+1} = A y_t + \kappa_t$. However, there are always perturbations so that we put

$$x_{t+1} = A y_t + \xi_{t+1} . \quad (3.15)$$

We will use this equation also for the case that the robot is moving on the ground or embedded into the chain. In this case the “noise” term ξ has to account for all effects due to the coupling with other robots, the slip and friction on the ground, inertia effects, collisions, and the like. Fig. 3.6 gives an exemplification of this situation in an experiment with a realistically simulated robot.

In a closed loop approach without internal states, the controller is given in terms of the sensor value x_t as

$$y_t = K(x_t) . \quad (3.16)$$

In the simple case we use

$$K(x) = \tanh(Cx) \quad (3.17)$$

where the hyperbolic tangent (\tanh) confines the motor values between -1 and 1 , see Fig. 3.7, so that the wheel can rotate forward and backward. We may interpret this as a simple rate-coded neuron with \tanh activation function and synaptic strength C , the neural network realization, see Sect. 3.6, of the controller consisting just of this one neuron.

Using Eq. (3.16) in Eq. (3.15) leads to the following dynamics of the sensorimotor loop

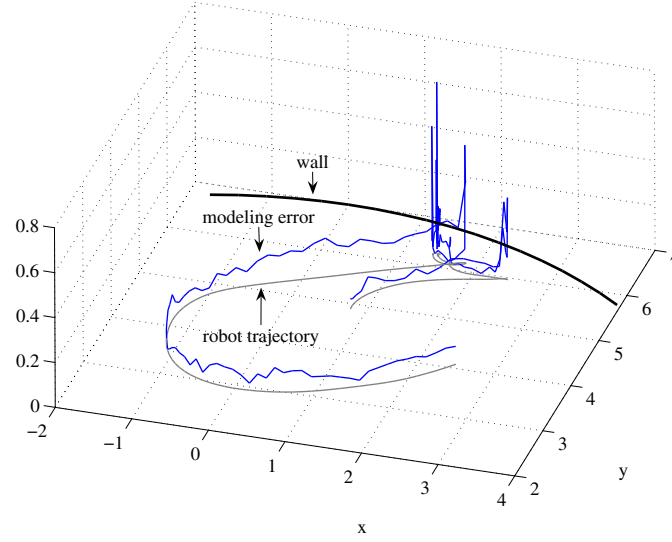


Fig. 3.6: Model error of a wheeled robot. A typical trajectory of the robot with the model errors ξ^2 , see Eq. (3.18). While during the normal drive the error is essentially given by the measuring uncertainties, the error increases drastically at the repeated collisions with the wall since wheels get blocked or may also rotate freely if the robot gets jammed. Control is realized by a homeokinetic controller as introduced later. Graphics taken from [68].
© MDPI Publishing and Frank Hesse.

$$x_{t+1} = A \tanh(Cx_t) + \xi_{t+1} \quad (3.18)$$

so that the map ψ of Eq. (3.8) is

$$\psi(x) = A \tanh(Cx) .$$

It will prove convenient to use also the formulation of the dynamics in terms of the membrane potential $z_t = Cx_t$ of the neuron where $R = CA$

$$z_{t+1} = R \tanh(z_t) + \rho_{t+1} \quad (3.19)$$

and

$$\rho = C\xi \quad (3.20)$$

is a modified noise variable.

3.3.1 Feedback Strength

The sensorimotor loop is a feedback loop, the (linear) feedback strength being given by

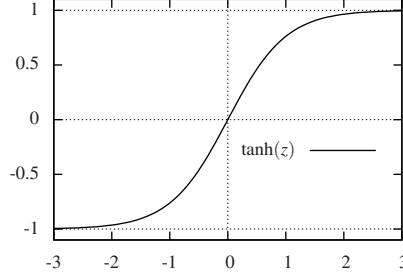


Fig. 3.7: Hyperbolic tangent squashing function. The range of $\tanh(z)$ is $(-1, 1)$. Used as an activation function the neuron has a linear response around $z = 0$ with slope 1. For large absolute z the neuron is in its saturation region with little input dependence.

$$R = CA .$$

The effect of R can be seen by the following argument. We consider the case of small z where the approximation $\tanh(z) \approx z - \frac{z^3}{3}$ can be used. Using the linear term only, one gets (ignoring the noise for the moment) from either Eq. (3.18) or Eq. (3.19)

$$x_t = R^t x_0 \quad (3.21)$$

meaning that the wheel slows down for $0 < R < 1$ ($x_t \rightarrow 0$ as $t \rightarrow \infty$) and accelerates for $R > 1$ (x_t increases exponentially), but the velocity can not increase unrestrictedly since $|\tanh(z)| < 1$, i. e. the nonlinearities confine the further growth of x_t .

3.3.2 Fixed Points

The asymptotic value of x in the nonlinear case is given by the solution of the fixed point equation obtained from Eq. (3.19) as

$$z = R \tanh(z) , \quad (3.22)$$

which has always $z^* = 0$ as a fixed point, which is stable for $0 < R < 1$ and unstable for $R > 1$. For $R < 0$ the state changes signed every iteration and is thus not useful for our purposes. At $R = 1$ there is a pitchfork bifurcation with two stable fixed points emerging. The stable fixed points can be found for instance by graphically solving the transcendental equation as shown in Fig. 3.8(a).

Another way is to use the approximation $\tanh(z) \approx z - \frac{z^3}{3}$ in order to get results for the case that $|z|$ is not too large. This is supported by the well known fact from dynamical systems theory that a system given by Eq. (3.19) is topologically equivalent [165] to the more simple system

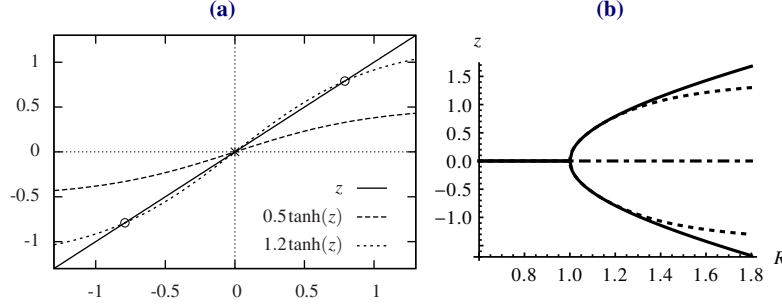


Fig. 3.8: Pitchfork bifurcation. (a) The graphical solution of $z = R \tanh(z)$ is exemplified for $R = 0.5$ with the stable fixed point at $z = 0$ and for $R = 1.2$ with two stable fixed points at $z = \pm 0.79$, the fixed point at $z = 0$ being unstable; (b) The bifurcation diagram is showing the fixed points in dependence on R . The **solid** line indicates the stable fixed points of the exact system and the **dashed** line of the approximation Eq. (3.23). The **dash-dotted** line marks the unstable fixed points.

$$z_{t+1} = R \left(z_t - \frac{z_t^3}{3} \right) + \rho_{t+1}.$$

In this way we may get a qualitative analysis of the system in simpler terms. This point will be discussed again in the gradient descent formulation of the dynamics, see Sect. 3.3.3.

Using this simplification, we obtain for $R = 1 + \delta$ with $0 < \delta \ll 1$ the two stable fixed points as (setting $\rho = 0$)

$$z = \pm \sqrt{3(R-1)} + O(\delta^2). \quad (3.23)$$

The approximation is valid only for $0 < \delta \ll 1$. The expansion in next order of the noise strength is given in Sect. 3.A.2.2 (p. 57), see Eq. (3.50). Further analysis of such a system (self-coupled single neuron) without noise is given in [121]. In [122] it was shown that such a neuron can even exhibit chaotic dynamics, however, not for the parameters used here.

3.3.3 Dynamics as Gradient Descent

The above derived properties of the sensorimotor loop can be seen more simply if rewriting the system dynamics as a gradient descent. This is possible for any one-dimensional system. In the present case we use

$$\frac{\partial}{\partial z} \ln \cosh z = \tanh z$$

to introduce a potential

$$V(z) = -R \ln \cosh z + \frac{z^2}{2} \quad (3.24)$$

so that Eq. (3.19) can be rewritten as

$$\Delta z_t = -\frac{\partial}{\partial z_t} V(z_t) + \rho_{t+1}. \quad (3.25)$$

As usual the gradient dynamics of Eq. (3.25) may be visualized by that of a sphere sliding down on the walls of a vessel filled with a viscous fluid.

The potential picture allows the discussion of the properties in simpler terms. For instance, the potential

$$V(z) = \gamma z + \alpha z^2 + \beta z^4 \quad (3.26)$$

with $\alpha = \frac{1}{2}(1 - R)$, $\beta = \frac{1}{12}R$, and $\gamma = 0$, corresponds to the third order approximation of the tanh function. It is easily seen that the potential differs from the full potential by a smooth deformation (without introducing new extremes) so that the qualitative properties of the gradient dynamics are preserved. In this way the notion of topological equivalence of dynamical systems gets a clear graphical demonstration, see Fig. 3.9. The above potential is well known from many branches of physics and dynamical systems theory.

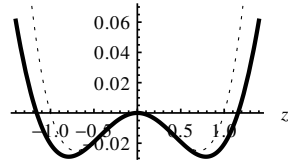


Fig. 3.9: Double well potential with approximation. Comparison of the exact potential, see Eq. (3.24) (**solid** curve), with its z^4 approximation Eq. (3.26), both for $R = 1.2$.

3.3.4 The Effective Bifurcation Point

If the noise is not zero it will not only modify the gradient step in size but may even change its sign. As a result, the system does not converge to the fixed point but instead fluctuates around the latter, provided the noise is weak and/or the fixed point has a high stability. In the bistable situation the noise may even drive the state over the potential barrier by a longer sequence of sign inverted gradient steps. Intuitively, the probability of such a transition will decrease exponentially with the length of an appropriate sequence which, at its hand, must be the longer the higher the potential wall and the weaker the noise.

Interestingly, the average time between such transitions is given [55] as a function of the strength D of the noise ρ Eq. (3.25) as

$$T_{\text{cross}} \propto e^{\frac{\Delta V}{D}} \quad (3.27)$$

where ΔV is the height of the potential barrier. The probability of crossing is close to one if $D \gg \Delta V$, i. e. if the potential barrier is low as compared to the fluctuations produced by the noise. On the other hand, if $\Delta V \gg D$, there is no barrier crossing at all in any physical time. There is a certain region where the system still feels the bistability but the barrier crossing is nevertheless substantial. In this region, the system, so to say, is already taking decisions (choosing one of the two fixed points) but is still very sensitively reacting to perturbations (e. g. by the environment) by revising decisions.

Right in that region we may define a point, called the *effective bifurcation point*², where the number of barrier crossings becomes critical in a relevant physical time. In the robotics application this will be the time scale of the behavior so that barrier crossing events will directly influence the behavior of the robot.

The concrete position of that point is empirical and can be defined by specifying the crossing time. In view of the exponential behavior of the crossing time, see Eq. (3.27), we better define the position of the effective bifurcation point by specifying the quantity (which is the logarithm of the crossing time)

$$\chi = \frac{\Delta V}{D} . \quad (3.28)$$

Details may be found in Sect. 3.A.2. The idea is illustrated in Fig. 3.10, which demonstrates that the effective bifurcation point is rather sharply defined even if the noise is rather strong. This is a direct consequence of the exponential dependence on the noise strength, see Eq. (3.27). Moreover, Fig. 3.10 shows that the effective bifurcation point reveals itself rather sharply if looking at the time averages of the noisy trajectories.

We can even find an analytical expression for the effective bifurcation point. As shown in the Appendix, see Sect. 3.A.2 (p. 56) Eq. (3.52), the effective bifurcation point is for small noise strength D given by the expansion

$$R_{\text{eff}} = 1 + \frac{2}{\sqrt{3}} \sqrt{\chi D^*} + \frac{22}{15} \chi D^* + O\left(D^{\frac{3}{2}}\right) \quad (3.29)$$

where $D^* = D/A^2$. With infinitesimal noise, the effective bifurcation point deviates from the deterministic one (which is at $R = 1$) by a square root law in the variance D of the noise. However, since this is an expansion in terms of \sqrt{D} , the next order terms rapidly come into play as the noise increases. Figure 3.11 confirms Eq. (3.29) with $\chi = 3.5$ in a convincing way. The idea is also illustrated by Fig. 3.12 displaying the probability distributions of the system state with different values of R . At the bifurcation point of the noise-free system ($R = 1$) a broad unimodal distribution is observed. For larger values of R a clear bimodal structure appears, and eventually

² The notion should not be confused with the stochastic bifurcation point, an abstract mathematical concept introduced by Ludwig Arnold [5]. In contrast to our effective bifurcation point the stochastic bifurcation is obtained from asymptotic limits. In our pitchfork bifurcation setting the bifurcation structure would not be revealed. Moreover, asymptotic limits are not useful in robotics due to finite behavior timescales.

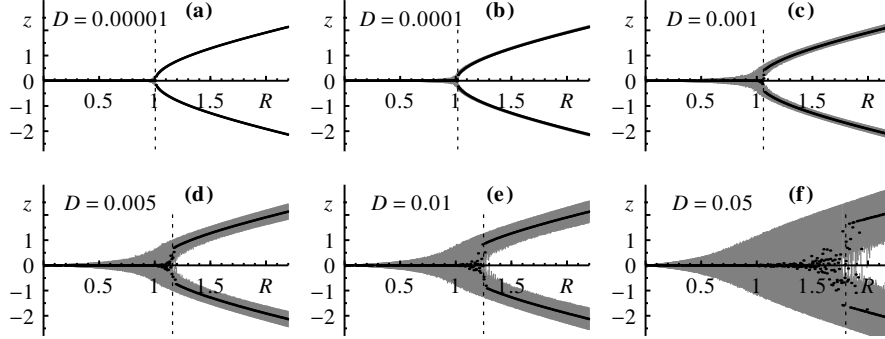


Fig. 3.10: Effective bifurcation diagrams for different noise strengths. The variance D of the uniform distributed noise is increased from panel (a) to (f). The **gray** area marks the hull of all trajectories (for each value of R 5000 iterations with positive and negative initial value). The **black** dots mark the mean value of 20000 iterations for each value of R and the **dashed** line shows the effective bifurcation point determined by eye.

the probability for the state to be in the region around $z = 0$ decays exponentially to zero.

We will encounter many illustrative examples showing that the effective bifurcation point defines a working regime, where systems are particularly avid to self-organize. First examples will be given in the following.

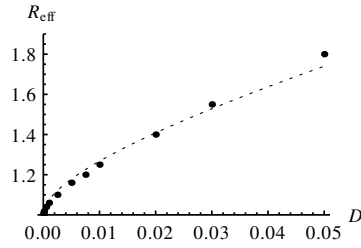


Fig. 3.11: Effective bifurcation point in dependence on the noise strength.

The **dashed** line shows Eq. (3.29) with $\chi = 3.5$ and the **black** points show the data taken from the simulations (partly displayed in Fig. 3.10).

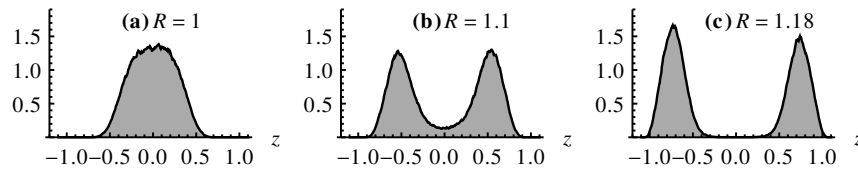


Fig. 3.12: Stationary probability distributions of the states (z) for different values of R . In all panels $D = 0.005$, see Fig. 3.10(d). (a) $R = 1$ (bifurcation point of the noise-free system); (b) $R = 1.1$ (R_{eff} for $\chi = 1.24$); (c) $R = 1.18$ (R_{eff} for $\chi = 3.5$).

3.3.5 Effective Bifurcation Point and Explorative Behavior

The effective bifurcation point is a novel concept with many interesting and far reaching consequences for the closed loop control of embodied robots. Let us consider a few examples, starting with a two-wheeled robot with each wheel being controlled individually and the sensor measuring the actual wheel velocity. We use in all examples the above controller with parameter C defining the feedback strength $R = AC$, using $A = \mathbb{I}$ for simplicity in the following. In the applications, the noise may be very strong and of varying nature, thinking for instance of a collision with an obstacle as in Fig. 3.6, so that the above requirements for the validity of Eq. (3.29) are not guaranteed. Nevertheless the concept of the effective bifurcation point remains valid. In fact, numerical experiments support that there is a relatively sharp value C_{eff} defining a favorable working regime in the sense explained above. You may convince yourself by doing Experiment 3.3.

Experiment 3.3: Closed loop control of wheeled robots.

For this experiment you have two choices: (a) a **single wheeled robot** in a square arena and (b) a **chain of robots**. The simulations start with the coupling set to $C = 1.2$ and the noise strength set to $\text{noise} = 0.1$ ($D = 0.01$). Decrease the coupling constants by entering:
`>coupling=value`
 until the behavior starts to change frequently, which should happen around $C_{\text{eff}} = 1.05$. This is especially visible in simulation (b), where below C_{eff} the chain barely moves. In order to exert external forces to the robots use either `<Ctrl>+<left Mouse button>` or `<Ctrl>+<right Mouse button>`. You can add/remove random obstacles by pressing `o` or `O` in all simulations.

The behavior of the robot is in good agreement with the message from the effective bifurcation diagrams of Fig. 3.10 and Fig. 3.12. With values $C < C_{\text{eff}}$ the wheel velocity is essentially fluctuating around zero, so that the robot executes a very slow random walk. The amplitude of the fluctuations is increasing rapidly if C approaches C_{eff} . With $C \gg C_{\text{eff}}$ the state is caught essentially in one of the fixed points so that the wheel velocity is fluctuating around this fixed point. Then, the robot is in one of four states, moving either forward or backward or rotating on site clockwise or counterclockwise.

A more favorable behavior occurs if C is close to C_{eff} since then there are occasional random transitions between these four states so that the robot realizes a rather effective exploration of the space. The nature of this exploration sensitively depends on the value of C since it defines the frequency of the switching between the states.

3.3.5.1 Robot in Cluttered Environments

The benefits of this closed loop control mode reveal themselves most clearly if the robot is colliding with an obstacle. In collision events, the wheels can get blocked

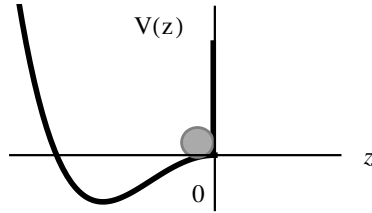


Fig. 3.13: The collision with an obstacle in the potential picture of the dynamics. The robot was moving forward, i.e. the state was at the r.h.s ($z > 0$) minimum of the double well potential. The impenetrable object corresponds to an infinitely steep rise in the potential so that the state is bound to move to the left minimum and the robot starts moving backward.

or at least slowed down³. In either case the **effective** feedback strength of the loop decreases so that z and with it the output y of the neuron decays. The only activity in the loop then comes from noise events, either by the measurement noise or the actuator noise (there are many different possibilities for a jerkily motion of the wheels, see the experiment). These fluctuations will be amplified if they are of the right sign, i.e. if the robot is momentarily moving away from the obstacle since then the wheels get free and the velocity can grow due to the still positive feedback. Hence, after a (short) sequence of collision events the state of the system goes to the other fixed point so that the robot is moving away from the obstacle, see Fig. 3.13 and Fig. 3.14. We may say that in this elementary sense the robot ‘feels’ the obstacles and generates a behavior away from them after a collision. There are no bumper sensors necessary for this to happen. Instead the behavior is an emerging property of our specific control mode close to the effective bifurcation point. You can verify this by doing Experiment 3.3 with a single robot.

The experiments reveal also another important property, namely that, once the initial hard collision has taken place, the probing of the obstacle by collisions happens in a quite gentle way so that the risk of damages is reduced. This is also an immediate consequence of the dynamics under the effective bifurcation regime, see Fig. 3.14.

As a consequence of this ‘feeling’ for the body, the robot will explore its environment in an effective way also in the case of a heavily cluttered arena. In Sect. 3.5 we will discuss how the robot can get this ‘feeling’ of the body without collisions by a virtual extension of the body using proximity sensors.

3.3.5.2 Emerging Cooperativity in a Chain of Robots

The sensitivity towards external influences produces an interesting effect if we consider a chain of passively connected robots (via ball-joints) with each wheel being controlled individually by a controller of the above kind. The behavior depends in an even more sensitive way on the value of the feedback strength in the loops. The region around the effective bifurcation point is again that of the most interesting behaviors. There the wheel velocities can become already quite large (decisions are taken) but can easily be switched by (i) noise events caused for instance by collisions

³ The wheel velocities may also change sign due to an elastic collision, in which case the robot will automatically move away from the obstacle.

with obstacles or (ii) by the influence of the forces exerted by the other robots in the chain. In fact, due to the high sensitivity, the wheel velocity will have a tendency to switch sign if a torque in the opposite direction is exerted by the other robots in the chain. By switching the velocity, the wheel is now acting in the direction of the force exerted on it. This self-amplification effect is essentially what is necessary for a self-organized synchronization of the wheel velocities.

Video 3.3 demonstrates quite clearly the strength of this self-organized synchronization effect, which not only makes the robot chain move into one direction but also keeps it still explorative in the sense that, after some time, it spontaneously inverts its direction of motion. Moreover, when colliding with an obstacle, the chain of robots often will change velocity in an integrated manner. Finally and most importantly, it will also effectively explore the spatial extensions of a maze, see Video 3.4 and/or do Experiment 3.3 with a chain of robots.

In order to demonstrate this point in a more general context, we have carried out a series of experiments with a more complex controller. Instead of using a controller for each wheel, each robot has now a controller with two motor neurons each of, which receiving the inputs from the two wheel sensors. Thus, there is a 2×2 coupling matrix C that we parameterize as

$$C = \begin{pmatrix} c & b \\ b & c \end{pmatrix}$$

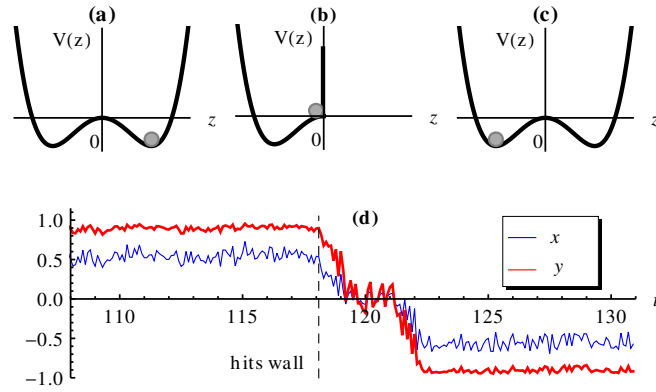


Fig. 3.14: Robot at a wall. (d) Neuron output y and wheel velocity sensor x of the closed loop control system (with feedback strength $R > R_c$) before and after a collision with a wall as obtained from an experiment with a real Khepera robot. Note that the wheels of the robot may slip, such that the activity in the loop is slowly decaying at the wall contact. Panels (a-c) display the corresponding potential $V(z)$ the sphere marking a stationary state (fixed point) of the system. Before the collision the system is in the fixed point with positive sign (robot is driving forward). During the time 118 to 122 the robot is kept at the unstable fixed point $z = 0$. At around 121.5 sec the robot starts moving backward because of the noise amplification effect.



Video 3.3: Emerging cooperativity in a chain of TwoWHEELS. The arena has no obstacles. Control is completely decentralized, but the individual wheels spontaneously cooperate in the working regime close to the effective bifurcation point. The video can be watched at <http://playfulmachines.com>.



Video 3.4: The chain of robots with decentralized control in a regular maze. Spontaneous cooperativity and sensitive reactions to collisions helps the chain to navigate in the maze without any proximity sensors. The video can be watched at <http://playfulmachines.com>.

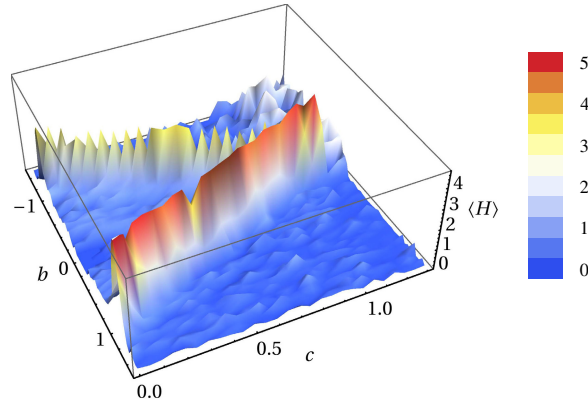


Fig. 3.15: Exploration of a maze by the chain quantified by the spacial entropy. The average spacial entropy $\langle H \rangle$ is depicted in dependence on the coupling parameters c and b . It is maximal if all sites are visited by the robot with equal probability and it is zero if the robot remains in its starting position. Interestingly, the coverage of the maze is best for the decentralized controller setting with $b = 0$, see [40] for details.

in accordance with the symmetries of the robot. The most essential point is that the robot chain explores the space effectively only for very subtle combinations of the parameters b and c . This is reflected by the spacial entropy (entropy of distribution of visited sites), see Fig. 3.15. It is only with those parameter values that the 10 wheels of the chain coordinate their activities so that the chain can act as a whole. In the two-dimensional parameter space, these combinations define a line of effective bifurcation points, see [40], which again shows the importance of that concept. Even more interestingly, the best exploration and hence the best spontaneous cooperation between the robots is obtained if $b = 0$, which means that the channels decouple or that each wheel has its individual controller. This is an interesting result in the direction of embodied robotics meaning that, due to the embodiment, higher coordination is achieved with less control, see [40, 191] for more details and the Videos 3.3 and 3.4.

3.4 Extending the Parameter Space

The analysis and results given so far have demonstrated that, under the closed loop setting, very rich behavior is observed even if the controller neuron has only a single parameter. This can be advanced further by including an additional parameter that acts as a bias for the neuron. As we will see, this extension of the parameter space creates a new behavioral feature—the system shows hysteresis. These properties are investigated as before by first analyzing the fixed point nature of the deterministic system.

3.4.1 Bifurcation Scenario.

When including a bias h into the controller, i. e.

$$K(x) = \tanh(Cx + h)$$

the fixed point equation of the deterministic system is, written in terms of the membrane potential, given by

$$z = R \tanh(z) + h. \quad (3.30)$$

Using the approximation of $\tanh(z)$ (Sect. 3.3.2), one gets from Eq. (3.30)

$$z = R \left(z - \frac{z^3}{3} \right) + h. \quad (3.31)$$

The stability and existence of the fixed points is well known so that we can draw upon the results obtained for such systems, see e. g. [121]. The bias leads to a cusp bifurcation where we have always one stable fixed point and additionally we observe

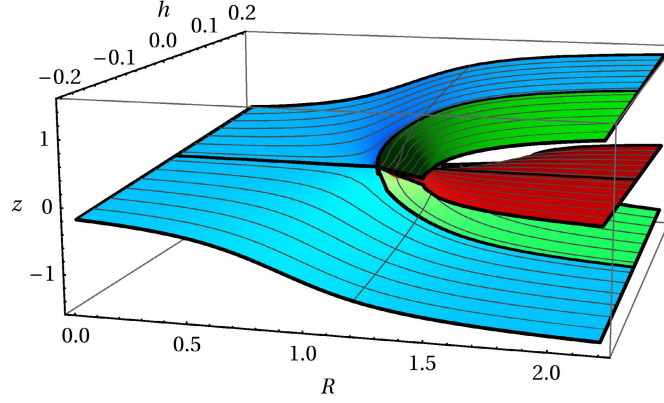


Fig. 3.16: Bifurcation diagram for the closed loop with bias. The diagram depicts the solution of $z = R \tanh(z) + h$, see Eq. (3.30), as a surface over R and h , that is called cusp bifurcation. Colors: *blue, green* are stable fixed points and *red* is unstable.

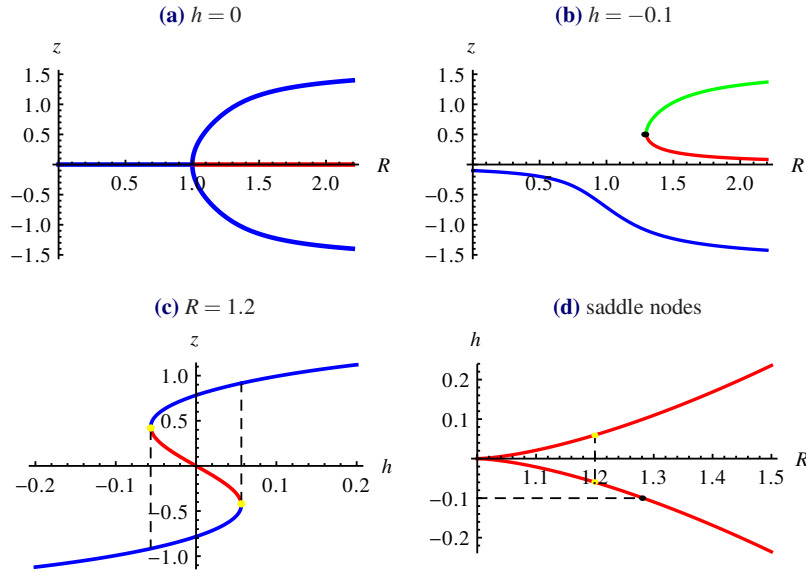


Fig. 3.17: Bifurcation diagrams for $z = R \tanh(z) + h$. The diagram depicts different cuts through the surface in Fig. 3.16. **(a)** Section at $h = 0$; **(b)** Catastrophic bifurcation at $h = -0.1$; **(c)** The hysteresis in dependence of h in the supercritical regime at $R = 1.2$, also indicated in Fig. 3.16. **(d)** Typical cusp wedge showing the saddle nodes in dependence of R and h . The colored points are correspondingly marked in **(b,c)**. Colors: *blue, green* are stable fixed points and *red* marks unstable fixed points.

the emergence of a so-called saddle-node bifurcation point, sometimes also called a catastrophic bifurcation because of the sudden appearance or disappearance of a pair of fixed points, one stable the other one unstable. A saddle-node bifurcation point occurs if the solution of the cubic fixed point equation (3.31) has three real solutions of which two are coinciding. The solution, is plotted as a bifurcation diagram for the parameters R and h in Fig. 3.16 showing a cusp bifurcation [2]. The saddle-nodes are located at the line where the red and green surfaces are intersecting.

Figure 3.17 shows different sections of the fixed point structure for better illustration. The positions of the saddle nodes form the typical cusp wedge, see Fig. 3.17(d). With $R < 1$, the system has only one stable fixed point. We call the system subcritical, since the fixed point is at small z and thus little activity occurs in the sensorimotor loop. For $h = 0$ we observe the same pitchfork bifurcation as before, see Fig. 3.17(a). For $h \neq 0$ two disconnected branches emerge, see Fig. 3.17(b), and the bifurcation becomes catastrophic, since the system state can undergo a drastic transition for small changes in parameter values. To illustrate the consequences, Fig. 3.17(c) displays the bifurcation diagram for the supercritical parameter $R = 1.2$ in dependence of h revealing a clear hysteresis effect. This means that the system resides in its fixed point when the parameter h is slowly decreased or increased until the fixed point disappears and the state jumps to the fixed point with the opposite sign (dashed lines). If the parameter h is changed in the other direction the same behavior is observed. Hence, for one parameter configuration the system can be in two possible states and h can be used to force a transition.

The visualization of the dynamics can again be done in the gradient descent picture, see Fig. 3.18.

3.4.2 Application: Biasing the Behavior

Hysteresis systems have interesting properties that have been exploited in many branches of science and technology. In our case, if we set the feedback strength a little above the critical point, a periodic signal of the bias with small amplitude is largely amplified by the dynamics of the sensorimotor loop. In the homeokinesis control mode to be discussed further below, the bias will be found to self-regulate. In that scenario, the amplification of the bias signals plays an essential role in the emergence of nontrivial behaviors.

The properties of the bias can immediately be used for biasing the behavior into a desired direction. For instance, in the simple two-wheeled robot experiment we can force the robot to move preferentially into the forward direction by choosing the same positive value of the bias for both the neurons. By choosing a bias of moderate size, we can preserve the reactions of our two-wheeled robot to collisions with some obstacle and nevertheless retain its forward preference, reducing the on-site rotation probability so that the exploration of the space will be enhanced.

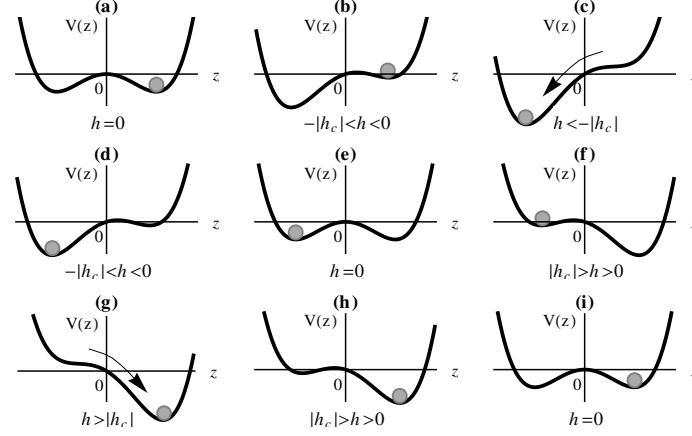


Fig. 3.18: The hysteresis cycle in the gradient picture. The diagrams show the stages of one hysteresis cycle starting from $h = 0$ (a) with the state at $z > 0$ as represented by the sphere. Decreasing h leads to a deepening of the left minimum, while the right minimum gets more flat, but the state remains at the minimum at $z > 0$, see (b). If $h = -h_c$ the saddle-node bifurcation happens, i. e. both the maximum at $z=0$ and the right minimum disappear so that the system shifts to the left minimum of the potential (c). Increasing h until $h = 0$ brings us back to the initial situation with the difference that the system is now at the fixed point with negative sign, see (d,e). The diagrams (f) and (g) show the switching from the minimum at $z < 0$ to the minimum at $z > 0$ by increasing h . By decreasing h until $h = 0$ the hysteresis cycle is finished, see (h,i).

3.5 Expanding the Body

In the scenarios considered so far, the robot can feel its environment only by collisions or other direct physical influences. This is of course annoying in practical applications where collisions might be dangerous. One way to help this is to virtually expand the body of the robot by exteroceptive sensors so that the collision can be felt before it takes place physically. A convenient choice are infrared sensors, which gradually change their sensor values as a function of the distance to an obstacle. Let us assume we equip the robot with six such sensors as depicted in Fig. 3.19. The output y_i of the controller for wheel $i \in \{\text{left, right}\}$ now is chosen as (dropping time indices, same C for both wheels)

$$y_i = \tanh(Cx_i + h_i)$$

where h_i is a bias given by a linear combination of the infrared sensor values s_j

$$h_i = \sum_j B_{ij} s_j \quad (3.32)$$

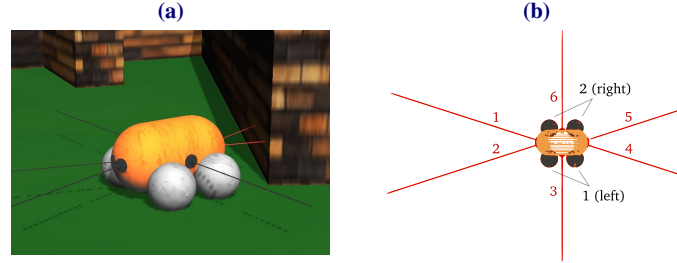


Fig. 3.19: Virtual expansion of the body. Infrared sensors, usually seen as exteroceptive sensors measuring distances to an obstacle, are considered here as defining a soft expansion of the body that is used as elastic collision sphere. The robot is the FOURWHEELED of the LPZROBOTS simulator that can be operated in a two-wheeled mode, meaning that the wheels at one side receive the same motor signal. The wheel velocity sensor reports the mean of both wheel sensors. (a) Screenshot taken from a simulation. The infrared (IR) sensors are drawn for illustration in *black* if they measure no obstacle and in *red* if they do; (b) Wire-frame view with IR sensor rays.

with a conveniently chosen set of parameters B_{ij} (typically in $[-1, 1]$) that determine the reactions of the robot to the proximity sensors.

The biasing effect can be understood in terms of the unbiased loop by introducing an effective sensor value x_{eff} , defined via $Cx_{\text{eff}} = Cx + h$. Changing h according to Eq. (3.32) acts like a variation of the sensor values x_{eff} just like in a collision. The effects of collisions can be mimicked if the B_{ij} are chosen such that a slowing down or switching of the wheel velocity is generated when approaching an obstacle. Moreover, by choosing C appropriately, the reactions of the robot can be made more smooth or more robust against the switching signals h_i , so that, by choosing the couplings B and C , a large number of behavioral patterns can be achieved. Experiment 3.4 shows that the robot will again explore its environment effectively, however now without colliding with the obstacles.

We see here some parallel to the famous Braitenberg vehicles [22], in that we can by simple means generate complex behaviors of the robot moving in an environment. It is not difficult to imitate some of the vehicles proposed by Braitenberg but we do not go into the details here. Altogether, the parallel is also in the general attitude of externalizing complexity from the internal control structure to the interaction with the environment.

Experiment 3.4: Expanding the body.

Start the simulation [Expanding the Body \(FourWheeled\)](#). Some obstacles are placed in the environment, you may add more with `o` and remove them with `O`. Try different coupling strengths for the IR sensors by entering:

```
>alpha=value
```

The default value is 1.0. The coupling matrix set to $B_{ij} = \alpha \begin{pmatrix} -1 & 0 & 1 & 0 & 1 & -1 \\ 0 & -1 & -1 & 1 & 0 & 1 \end{pmatrix}$, which leads to an obstacle avoidance behavior. You can also change each individual entry with the parameters B11-B26 to get a different behavior. Alternatively you can try [Expanding the Body \(LongVehicle\)](#) where only four infrared sensors are used.

One advantage of our setting is seen in the fact that the collision handling works even in the case that the obstacle is not seen by the IR sensors. In that case the physical collision will take place with the switching of the wheel velocities due to the blocking effect on the wheels as described above. Thus there is always a kind of minimal strategy for survival.

The remainder of this chapter is devoted to the introduction of neural networks that are used for the implementation of the control systems. It may be skipped and looked up as needed.

3.6 Realization by Neural Networks

The implementation of the parameterized functions for controllers and forward models are realized in practice in many different ways. A standard approach is to use artificial neural networks because they are universal, easy to implement, and have well structured algorithms for the adaptation of the parameters. Since we want to derive explicit expressions later on, we will introduce very briefly the main points of that approach in the following.

We will consider a network of rate-coding neurons, where the activity of each neuron at discrete time steps is represented by a single number, which can be interpreted as a mean firing rate. Actually, biological neurons have a very complicated internal dynamics, but in a simplified view we can think of a membrane potential that is driven by incoming spikes from other neurons. Once the membrane potential has reached a certain threshold, a single spike is fired and the membrane potential is reset. In this picture the time distribution of the spikes may carry information (time coding). This is known to play a role for instance in correlation learning via spike timing dependent plasticity (STDP) [36]. However, in many cases it was shown that neurons transmit information mainly via their firing rate (mean activity), which is especially prominent in coding of sensory stimuli. This is the so called rate-coding paradigm, which is used in the artificial neural networks (ANNs) we are dealing with. Viewed as information processing systems, ANNs are much closer to the brain than to a classical computer program. In particular the ANNs are prominent due to their ability to learn.

There are many excellent textbooks, e. g. [114, 167], dealing with the theory and application of artificial neural networks, especially in robotics, so that we will sketch here only the facts most relevant to our work.

3.6.1 Rate-Coding Neuron Model

Under the rate-coding paradigm, the neuron can be represented by a simple mathematical function. The neuron is considered as an input-output device. The input is represented by a vector $x \in \mathbb{R}^n$ and the output of a neuron i is written as

$$o_i = g(z_i)$$

where $g : \mathbb{R}^1 \rightarrow \mathbb{R}^1$ is the so called activation (or transfer) function and z_i is the post synaptic potential given by the weighted sum over the inputs as

$$z_i = h_i + \sum_{j=1}^n W_{ij} x_j$$

the W_{ij} being the synaptic efficacies of the i -th neuron connected to a neuron or sensor input j and h_i is a threshold or biasing term. We denote by $W_i \in \mathbb{R}^{n_i}$ the weight vector of neuron i . The function $g : \mathbb{R}^1 \rightarrow \mathbb{R}^1$ is typically a sigmoid function like

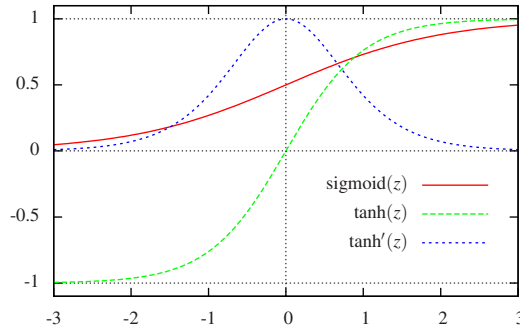
$$g(z) = \text{sigmoid}(z) = \frac{1}{1 + e^{-z}},$$

which monotonously increases from 0 to 1 as z increases from large negative to large positive values. We mainly use

$$g(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^{-z} + e^z} \quad (3.33)$$

mapping the input to the interval -1 to 1 , see Fig. 3.20.

Fig. 3.20: Neuron activation functions. Note the range of $\text{sigmoid}(z)$ is $(0, 1)$ whereas $\tanh(z)$ maps to $(-1, 1)$. The derivative of the hyperbolic tangent is given by $\tanh'(z) = 1 - \tanh^2(z)$.



3.6.2 Supervised Learning

A neuron i from above maps the inputs x to output o_i . The map is parameterized by the weight vector W_i . The weights can be adapted in the following learning scenario: given a set of input-output mappings (provided by a trainer) find W_i such that the neuron reproduces these pairs as close as possible. In an on-line learning scenario, the neuron sees an input x together with the target output o_i^{teach} , which is provided

by the trainer, in each time step. The error of the neuron is

$$E_i = \frac{1}{2} (o_i - o_i^{\text{teach}})^2 = \frac{1}{2} \delta_i^2 \quad (3.34)$$

where $o_i = g(z_i)$ is the current output of the neuron and $\delta_i = o_i - o_i^{\text{teach}}$ is the mismatch between current output and target output. Gradient descent on this error yields the update rule, also known delta-rule, for both the synaptic weights and the thresholds as

$$\Delta W_{ij} = -\varepsilon d_i x_j \quad (3.35)$$

$$\Delta h_i = -\varepsilon d_i \quad (3.36)$$

where

$$d_i = g'(z_i) (o_i - o_i^{\text{teach}}) = g'(z_i) \delta_i, \quad (3.37)$$

which can be considered as an error signal present at the output of the synapse W_{ij} . The structure of d_i is generic since it can be considered as the result of propagating the error signal δ_i at the output of the neuron through the activation function, thereby producing a factor $g'(z_i)$, backwards to the output of the synapse.

The update of W_{ij} is given by the product of this output signal multiplied by x_j , which is the activity at the input of the synapse. This is reminiscent of the general rule introduced by D. Hebb [66] who postulated that the synaptic strength increases if activities on the pre- and postsynaptic side are both high, often stated as “fire together—wire together.” However the delta-rule is still different from pure Hebbian learning. The difference of Eqs. (3.35–3.37) to Hebb’s rule is (i) that the postsynaptic activity is not that of the neuron but the error signal propagated back from the output of the neuron and (ii) that the update can be both positive and negative. Nevertheless we will find this parallel to Hebb’s rule very helpful also with our more complicated learning rules driving the self-organization process.

3.6.3 Feed-Forward Networks

Neurons can be combined to neural networks in order to represent more complicated input output mappings. In a typical feed-forward architecture the net realizes a function $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$, i.e. it maps inputs $x \in \mathbb{R}^n$ to outputs $o \in \mathbb{R}^m$. The network is organized as a sequence of layers, the neuron outputs of a layer forming the inputs to the neurons of the subsequent layer. A network with N layers consists of the input layer where x is fed in, a number of hidden layers, and an output layer of m neurons. Numbering the layers from the input layer ($k = 0$) to the output layer ($k = N$) the rule for feeding the input information through the network is given iteratively by

$$o_i^{(k)} = g \left(h_i^{(k)} + \sum_j W_{ij}^{(k)} o_j^{(k-1)} \right), \quad \text{for } k = 1, 2, \dots, N \quad (3.38)$$

(identical activation functions for all neurons) starting with $o^{(0)} = x$. The output vector of the network is given by $o^{(N)}$.

There is well known theorem stating the universality of neural networks as a function approximator: Already a network with only one hidden layer (with a sufficient number of neurons) is able to represents any function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with arbitrary accuracy see [33]. An intuitive explanation is that the neurons of the hidden layer can be organized into groups so that the collective receptive field of a group covers regions in input space mapping to about the same value of the target function f . Then, all the output layer has to do is to weight the contributions of the groups in order to produce the correct output $o^{(\text{teach})} = f(x)$ to any input x .

This function approximation can be learned by gradient descending the error

$$E = \left(o - o^{\text{teach}} \right)^2. \quad (3.39)$$

This can be transformed into a systematic procedure for the individual updates of the synaptic weights

$$\Delta W_{ij}^{(k)} = -\epsilon d_i^{(k)} o_j^{(k-1)}, \quad (3.40)$$

which is again Hebb like. The error activities are obtained in an iterative way

$$d_i^{(k)} = g' \left(z_i^{(k)} \right) \sum_l d_l^{(k+1)} W_{li}^{(k+1)}, \quad \text{for } k = 1, \dots, N-1 \quad (3.41)$$

starting with $d_i^{(N)} = g' \left(z_i^{(N)} \right) (o_i - o_i^{\text{teach}})$ as in the single layer case. The error signal for neuron i in layer $k < N$ is produced by the weighted sum of the error signals in the layer $k+1$ multiplied by the derivative of the gain function. This is the famous error backpropagation rule [180]. In matrix notation we write

$$d^{(k)} = G' \left(z^{(k)} \right) \left(W^{(k+1)} \right)^\top d^{(k+1)}, \quad \text{for } k = 1, \dots, N-1 \quad (3.42)$$

where G' is the diagonal matrix given by $G'_{ij} \left(z^{(k)} \right) = \delta_{ij} g' \left(z_i^{(k)} \right)$ with δ_{ij} being the Kronecker delta and W^\top being the transpose of a matrix W .

3.6.4 Recurrent Networks

The feed forward networks serve well the purpose of parametrized function approximation. However in many applications, in time series prediction e.g., the output of the network is depending on previously seen inputs. Another example is the realization of a controller with internal states, see Eq. (3.4). This can be achieved with additional inputs into the network generated from earlier events. A more elegant way consists in recurrent neural networks (RNN), which internally built up a memory of the past by time delayed recurrences.

A simple RNN, introduced by Elman [48], is obtained by equipping a feed-forward network with feedback loops providing a time delayed copy of the outputs of all neurons of the net, or a group thereof, back to the neurons itself. As before the inputs are presented to the network at times $t = 0, 1, \dots$. The iterative rule (3.38) for evaluating the output of the network is now (sum over repeated indices)

$$o_i^{(k)}(t) = g\left(W_{ij}^{(k)} o_j^{(k-1)}(t) + V_{ij}^{(k)} o_j(t-1) + h_i^{(k)}\right), \quad \text{for } k = 1, 2, \dots, N \quad (3.43)$$

where $o^{(0)}(t) = x_t$ is the current input into the network, $o(t-1)$ the vector of previous activations of the ensemble of neurons in the net, and V is a weight matrix controlling the feeding back of these activations. We are not going into any details here since we are introducing recurrences in a different way by the principle of homeokinesis, see Chap. 5.

The network is now a discrete-time dynamical system driven by the inputs x_t . Depending on the weights, the network can develop the full scope of dynamical behaviors ranging from fixed point attractors over limit cycles to chaos. Again the important point is that the weights can be learned such that the network reproduces a target dynamics. The learning rules are a bit more complicated but are systematic and well investigated. It is this property which makes the recurrent networks valuable tools for time series prediction, system identification, signal processing, and last but not least robot control.

3.7 Summary

To summarize, given a simple closed loop system with a single nonlinear neuron we find non-trivial fixed points and hysteresis depending on the control parameters. This type of dynamics is found in many systems, e. g. in statistical mechanical models of magnets, see [165]. We introduce the effective bifurcation point, since we consider the system with noisy perturbations. For robot control it seems suitable to keep the feedback strength R at the effective bifurcation point, slightly above the bifurcation point of the deterministic system, in order to have two stable fixed points allowing the system to switch either by changing the bias h or by external influences. In the case of multiple sensors and motors we get of course a much larger number of attractors, Neimark-Sacker bifurcations, limit cycles and so forth, which will be worked out in Chap. 7 and demonstrated by applications to robotic systems later in this book.

Appendix 3.A Mathematical Details

We give here the mathematical details for the stability analysis of the fixed points without bias and the evaluation for the position of the effective bifurcation point.

3.A.1 Stability Analysis

The properties of the fixed points are most easily obtained by a linear stability analysis. We put $z_t = z^* + u_t$ where u is small. The state dynamics $z_{t+1} = Rg(z_t)$ is linearized as

$$z^* + u_{t+1} = Rg(z^* + u_t)$$

and by using Taylor expansion one gets

$$u_{t+1} = L(z^*)u_t$$

where

$$L = Rg'(z^*)$$

is the derivative. One has only to show that $0 < L < 1$ in order to prove that $u_t \rightarrow 0$ as $t \rightarrow \infty$. Using the approximation Eq. (3.23) we get

$$L = Rg'(z^*) = R \left(1 - \frac{3(R-1)}{R} \right) = 3 - 2R$$

so $L < 1$ if $1.5 > R > 1$, provided $\tanh(z) = z - \frac{z^3}{3}$ is valid, i.e as long as R close to 1. However if $R \gg 1$ then $|z|$ is very large and one gets from the fixed point equation ($z = R \tanh(z)$) that $z^* \simeq R$, so $L = R(1 - \tanh^2(R))$ or approximately $L = 4R \exp(-2R)$, since at large R we may write approximately

$$\tanh R = \frac{e^R - e^{-R}}{e^R + e^{-R}} = \frac{1 - e^{-2R}}{1 + e^{-2R}} \approx 1 - 2e^{-2R}$$

Hence L goes exponentially to 0 as $R \rightarrow \infty$ so that the stability of the fixed point increases with increasing R .

These results have been derived in some detail since L is the central quantity of the time-loop error, as we will see later. We have seen here that it is directly related to the stability of the fixed points of the state dynamics. In the higher dimensional cases L is the Jacobian matrix of the dynamical system the eigenvalues of which define the stability of the system.

3.A.2 Determining the Effective Bifurcation Point

The aim now is to find an explicit expression for the position of the effective bifurcation point (EBP) in the case of the pitchfork bifurcation for the dynamics given by Eq. (3.19), i. e.

$$z_{t+1} = Rg(z_t) + \rho_{t+1} = -\frac{\partial}{\partial z_t} V(z_t) + \rho_{t+1}$$

The position of the EBP is defined by the empirical constant χ defining the relation between the barrier height and the variance of the noise as

$$\Delta V = \chi \langle \rho^2 \rangle \quad (3.44)$$

$\langle \rho^2 \rangle$ being the variance of the noise in the z dynamics.

3.A.2.1 Series Expansion of the EBP in the z^4 Case

In order to find ΔV we consider the z^4 potential case first, see Eq. (3.26)

$$V(z) = -\frac{R-1}{2}z^2 + \frac{1}{12}Rz^4,$$

where the maximum is given by $V(0) = 0$ and the minimum by $V(z^*)$ with

$$z^* = \pm \sqrt{3 \frac{(R-1)}{R}}$$

being either one of the stable fixed points so that the barrier height is

$$\Delta V = V(0) - V(z^*) = \frac{3}{4} \frac{(R-1)^2}{R}.$$

Eq. (3.44) reads now

$$\frac{3}{4} \frac{(R-1)^2}{R} = q. \quad (3.45)$$

where $q = \chi \langle \rho^2 \rangle$. The solution and its power series expansion (Taylor) is

$$R = 1 + \frac{2}{3}q + \frac{2}{3}\sqrt{3q+q^2} = 1 + \frac{2}{\sqrt{3}}3\sqrt{q} + \frac{2}{3}q + O\left(q^{\frac{3}{2}}\right)$$

In order to get an expression in terms of the original noise strength $D = \langle \xi^2 \rangle$, i. e. the one featuring in the x dynamics, we use $\langle \rho^2 \rangle = \langle \xi^2 \rangle C^2 = DR^2/A^2 =: D^*R^2$ so that now $q = \chi D^*$ and we have to solve

$$\Delta V = qR^2. \quad (3.46)$$

This equation is of third order in R so that it can be solved explicitly. The solution however is quite complex so that we better aim at establishing a series expansion in terms of q since we are mainly interested in the low noise case anyway. We try a power series expansion for R in terms of \sqrt{q} as

$$R = 1 + m\sqrt{q} + nq + pq^{\frac{3}{2}} + O(q^2) \quad (3.47)$$

and expand the condition given by Eq. (3.46) in terms of q as

$$\frac{3}{4} \frac{(R-1)^2}{R} - qR^2 = \left(\frac{3}{4}m^2 - 1 \right) q + \left(\frac{3}{2}mn - \frac{3}{4}m^3 - 2m \right) q^{\frac{3}{2}} + O(q^2) \quad (3.48)$$

where now $q = \chi D^*$. If the expansion Eq. (3.47) is to be a solution of Eq. (3.46), the coefficients in Eq. (3.48) must be zero in all orders that are relevant for the expansion of the solution. Up to order $q^{\frac{3}{2}}$ the coefficients are equal to zero if $m = 2/\sqrt{3}$ and $n = 2$ so that we get the result

$$R = 1 + \frac{2}{\sqrt{3}}\sqrt{q} + 2q + O(q^{\frac{3}{2}}).$$

Putting the results together, the low noise expansion of the EBP in the z^4 case is

$$R_{\text{eff}} = 1 + \frac{2}{\sqrt{3}}\sqrt{\chi D^*} + 2\chi D^* + O(D^{\frac{3}{2}}) \quad (3.49)$$

with $D^* = \frac{D}{A^2}$.

3.A.2.2 Series Expansion of the EBP in the tanh Case

The expression of the fixed point (FP) in the case of infinitesimal noise has been given already in Eq. (3.23). In order to get an expansion for the EBP comprising the next order we have to find first the corresponding expansion of the FP. We put $R = 1 + u$ with $0 < u$ and try $z = m\sqrt{u} + nu + pu^{\frac{3}{2}} + O(u^2)$ in the expansion of the FP condition $z = R \tanh z$ obtaining

$$R \tanh(z) - z = \alpha u^{\frac{3}{2}} + \beta u^2 + \gamma u^{\frac{5}{2}} + O(u^3)$$

where $\alpha = \left(m - \frac{m^3}{3}\right)$, $\beta = (n - m^2n)$, and $\gamma = \frac{2}{15}m^5 - \left(\frac{m^3}{3}\right) - mn^2 + p - m^2p$. The coefficients α , β , and γ are shown to be zero if $m = \sqrt{3}$, $n = 0$, and $p = \sqrt{3}/10$ so that the two stable FPs have the expansion

$$z^* = \pm \left(\sqrt{3u} + \frac{\sqrt{3}}{10}u^{\frac{3}{2}} \right) + O(u^{\frac{5}{2}}). \quad (3.50)$$

The potential now is, see Eq. (3.24),

$$V(z) = \frac{1}{2}z^2 - R \ln(\cosh z)$$

with the height of the barrier obtained as

$$\Delta V = 0 - V(z^*) = \frac{3}{4}u^2 - \frac{3}{20}u^3 + O(u^4)$$

The EBP is found by using for u the expansion $u = m\sqrt{q} + nq + sq^{\frac{3}{2}} + O(q^2)$ and $R = 1 + u$ in the condition $\Delta V = \chi D^* R^2$ so that

$$\frac{3}{4}u^2 - \frac{3}{20}u^3 - qR^2 = \left(\frac{3}{4}m^2 - 1\right)q + \left(\frac{3}{2}mn - 2m - \frac{3}{20}m^3\right)q^{\frac{3}{2}} + O(q^2) \quad (3.51)$$

the first two coefficients being zero if $m = \frac{2}{3}\sqrt{3}$ and $n = \frac{22}{15}$ so that for small noise strengths we obtain the solution of Eq. (3.45) for R_{eff} by the expansion

$$R_{\text{eff}} = 1 + \frac{2}{\sqrt{3}}\sqrt{\chi D^*} + \frac{22}{15}\chi D^* + O(D^{\frac{3}{2}}) \quad (3.52)$$

with the leading square root dependence for very small D as in the case of the z^4 potential.

The Playful Machine

Theoretical Foundation and Practical Realization of
Self-Organizing Robots

Der, R.; Martius, G.

2012, XX, 335 p., Hardcover

ISBN: 978-3-642-20252-0