TUTORIAL

Playful Machines

Theoretical Foundation and Practical Realization of Self-Organizing Robots

Ralf Der and Georg Martius
ralfder|martius@mis.mpg.de
Max Planck Institute for Mathematics in the Sciences
Leipzig, Germany

September 30, 2010

This is a preliminary chapter of a book that is to be published by Springer in 2011. It may be used as a tutorial to understand the basics of our work on self-organizing robots. The bibliography is very incomplete. For further reading we recommend the following papers. The principle of homeokinesis was introduced in the context of robotics for the first time in [9], it was further developed in [3] and [6] where the time loop error was introduced. This concept was generalized and applied to various robotic systems, cf. [4, 5, 7, 8]. Most recent developments are devoted to guiding self-organization by external cues, cf. [11, 12].
The videos referred to in the text are located at http://playfulmachines.com. The experiments proposed in the text can be carried out using our simulator. Please contact us by email for the password to download (for free).

This document will give the reader the opportunity to get an overview of our systematic approach to the self-organization of behavior without going into the details or using much mathematics. The method will be explained by the example of a comparatively simple machine which, however, has the advantage to elucidate the specific features and in particular the role of embodiment in our approach. Throughout the document the reader is encouraged to underpin the theoretical developments by conducting experiments with the simulator and in this way will get an deeper understanding of the method and also a training in using the simulator.

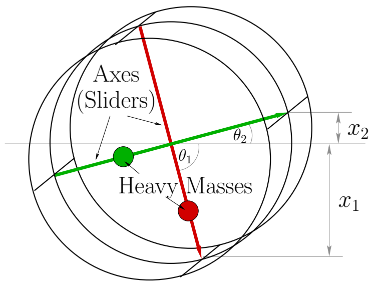


Figure 1: The BARREL. It consists of a cylindrical encasement with two weights sliding on two axes perpendicular to the cylinder axis. Each of the weights is moved by a linear motor so that the system can shift its center of gravity. The sensor values measure only the inclination of the axes, i. e. $x_i = \sin(\theta_i)$. Thus, the true physical state of the system is largely unknown to the controller. Note that $x_1 < 0$ in the displayed situation.

1 Dominated by Embodiment: The BARREL

The focus of our work is on the self-organization of control of robotic systems by exploiting as much of the embodiment effects latent in these systems as possible. While our emphasis lies on doing this with highly complex systems we will, in the current document, present the essential points at a level where transparency and complexity may meet. The machine we are going to use for is the BARREL (which is short for barrel robot), see Figure Fig. 1. The BARREL can deploy many different kinds of motion, despite its simple construction, as will be seen below.

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 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[14]. The internal weights are moved by a simulated linear motor with realistic properties, in particular a limited maximal force. Each bare motor is supported in doing its task by an extra PID controller which compensates for both overshooting and undershooting in the movement of the weights. Nevertheless, the motors can not move the weights with arbitrary velocity so that 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.

The dominance of these embodiment effects and the lack of knowledge of the dynamical state of the system (due to the poor sensor situation) makes the more traditional approaches to robotics difficult to apply. On the other hand, it will be seen that, under a closed loop control paradigm, very simple controllers are sufficient to support specific modes of behavior if they are, so to say, natural for the body, rolling being just one example. The aim of self-organizing robot control is to make these modes emerge in a natural way.

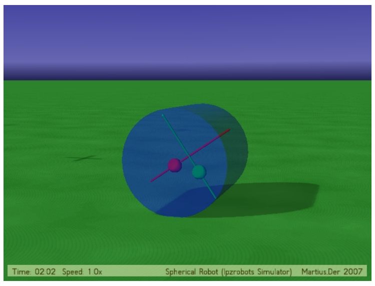
Another reason for choosing the BARREL as a demonstration object may be seen in its remote similarity to the famous passive walker [13] when driven by some periodic motor forces so as to be able to walk on a horizontal plane [2]. This walker works by having its center of gravity a little ahead of the 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.

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 AI 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.

1.1 Open Loop Control

The most simple way to control the BARREL is by just sending motor signals generated by a kind of central pattern generator. Let us first assume that the BARREL is to roll with a fixed velocity. This is achieved by shifting the internal weights periodically with a convenient frequency and phase shift $\pi/2$, the velocity of the BARREL being determined by that frequency – one rotation of the BARREL corresponds to one period of the pendular oscillation.

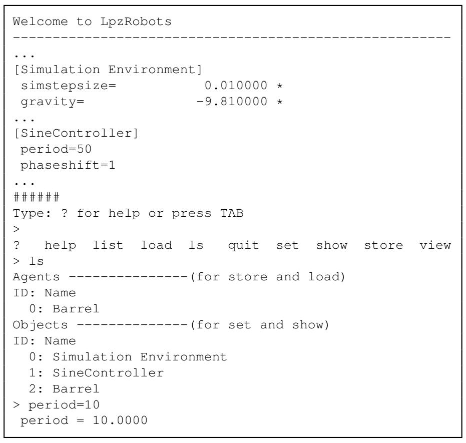
When doing so we find that, at low frequencies and with some friction at least, the BARREL adapts its (average) rotational frequency to the frequency of the pendular oscillations, see Video 1.1.



Video 1.1: The BARREL controlled by a periodic control signal with a phase shift of $\pi/2$ between the internal axes. 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 of the motors to execute the actions (periodic motions of the weights). The videos can be found at `http://playfulmachines.com`.

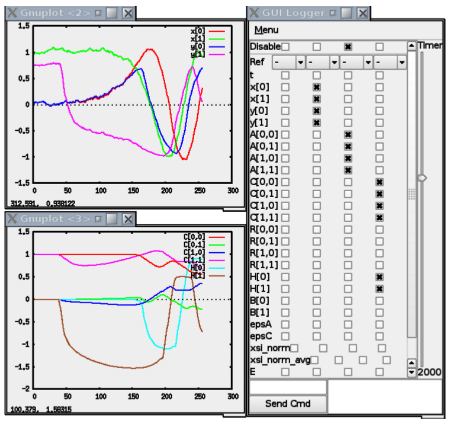
This behavior is stable against perturbations. To understand this, imagine a stroboscopic mapping depicting the BARREL every moment the red weight, say, has maximal downward elongation. In an ideal and constant rolling mode the red weight will be exactly below the cylinder axis since then there is no 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

(a)



```
Welcome to LpzRobots
-----------------------------------------------------------
...
[Simulation Environment]
 simstepsize=           0.010000 *
 gravity=              -9.810000 *
...
[SineController]
 period=50
 phaseshift=1
...
######
Type: ? for help or press TAB
>
?   help  list  load  ls   quit  set  show  store  view
> ls
Agents ---------------(for store and load)
ID: Name
  0: Barrel
Objects --------------(for set and show)
ID: Name
  0: Simulation Environment
  1: SineController
  2: Barrel
> period=10
 period = 10.0000
```

(b)



Disable	☐	☐	☒	☐
Ref	-	-	-	-
t	☐	☐	☐	☐
x[0]	☐	☒	☐	☐
x[1]	☐	☒	☐	☐
y[0]	☐	☒	☐	☐
y[1]	☐	☒	☐	☐
A[0,0]	☐	☐	☒	☐
A[0,1]	☐	☐	☒	☐
A[1,0]	☐	☐	☒	☐
A[1,1]	☐	☐	☒	☐
C[0,0]	☐	☐	☐	☒
C[0,1]	☐	☐	☐	☒
C[1,0]	☐	☐	☐	☒
C[1,1]	☐	☐	☐	☒
R[0,0]	☐	☐	☐	☐
R[0,1]	☐	☐	☐	☐
R[1,0]	☐	☐	☐	☐
R[1,1]	☐	☐	☐	☐
H[0]	☐	☐	☐	☒
H[1]	☐	☐	☐	☒
B[0]	☐	☐	☐	☐
B[1]	☐	☐	☐	☐
epsA	☐	☐	☐	☐
epsC	☐	☐	☐	☐
xsi_norm	☐	☐	☐	☐
xsi_norm_avg	☐	☐	☐	☐
E	☐	☐	☐	☐

Figure 2: User interface of the LpzRobots simulator. **(a)** Terminal window with console interface. **(b)** 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.

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.

Let us try to make contact with the embodied AI approach at this point. The aim of the latter is to shift the computational load from the controller to the morphology and physical properties of the embodiment. In the above case this is possible due to the self-stabilization effect – a simple periodic signal generates a stable motion pattern in a complex physical object. In order to get a nice harmonic motion one has two options, either to change the wave form of the control signal 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 is one more or less sharp frequency where the motion is most harmonic. This is where the control will also need the least energy. On the other hand, in the sense of the embodied AI approach, given the frequency, one may adapt the physical parameters like the PID settings, the forces and pendular ranges, the mass of the cylinder and so forth in order to get a 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 1.1 or do the Experiment 1.1.

The new perspective that self-organization can bring into this paradigm is that instead of adapting a system to produce a specific behavior pattern in the first place we may obtain a system that finds out, by self-exploration, about the whole range of modes it is able to support by controllers of low complexity. These modes may then be viewed as the candidates for embodied AI realization of more complex behavior architectures.

Before giving the details of that approach we must introduce closed loop control.

Experiment 1.1: 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 [10]. Here we will describe the handling of the software in more detail. Follow the installation instruction and start the OUTLINE-DEMONSTRATOR. A terminal window opens and you get a list of simulations which can be selected by their associated number. Select "Open Loop" by typing

`1`<Enter>

Another window is opened that shows the rendered scene which we will call the graphical window, see Video 1.1. 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. 2(a) 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). 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 following line should show

```
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 parameters

`>noise=`Strength

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

Try noise = 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. 2(b).

1.2 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. The body takes over the command in some sense 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. 1.

Quite generally, we assume that the sensor values at time t are comprised in the vector $x_t \in \mathbb{R}^n$ where $x_t = (x_{t1}, \ldots, x_{tn})^\top$ for n sensors. 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 χ so that we have the very simple sensor dynamics

$$x_{t+1} = U(\chi)\, x_t\,,$$

where U is the rotation matrix

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

rotating a vector by the angle χ.

Under the closed loop control paradigm, the controller is a function $K : \mathbb{R}^n \to \mathbb{R}^m$ mapping the vector x_t of sensor values to the vector y_t of motor values, so that

$$y_t = K(x_t)\,.$$

In the BARREL case $y_t \in \mathbb{R}^2$ gives the nominal positions of internal weights on their respective axes. Controller can be very difficult, in particular they may contain internal states that modify the mapping depending on contexts. 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

$$K(x) = Cx + h, \tag{1}$$

where $h = 0$ for the moment and

$$C = cU(\phi) \tag{2}$$

with c defining the amplitude of the weight shifting. When using this controller we find very stable rolling modes as expected. We may push the BARRELor 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 1.2.

If C is a rotation matrix one observes very stable rolling modes with a frequency defined by ϕ, see Eq. (2). However, it is observed that ϕ is in general larger than the true rotation angle of the sensor vector in one time step χ. 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 in advance anything about the system itself.

In general, we may obtain quite complicated behaviors already with this controller when using different matrices C. In order to keep the actions (motor values) in bounds we introduce a smooth squashing function $g(u) = \tanh(u)$, that is applied component-wise and set

$$K(x) = g(Cx + h) \tag{3}$$

where $h \in \mathbb{R}^m$ is a bias introduced for greater generality. In this way, motor values are kept safely in the interval $(-1, 1)$. We may consider $K(x)$ also as a one-layer neural network with tanh-neurons, but the controller can easily be generalized by introducing more layers and/or internal recurrences, cf. Chapter **??**.

Experiment 1.2: Closed loop control of BARREL

Start the OUTLINE-DEMONSTRATOR and select "closed loop control". 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.

Let us now see what happens if the controller parameters are chosen randomly: One observes a variety of behavioral modes which are sometimes surprising given the extremely simple nature of the controller. The variety of modes is even larger if the bias values h are chosen randomly as well, see Experiment 1.2.

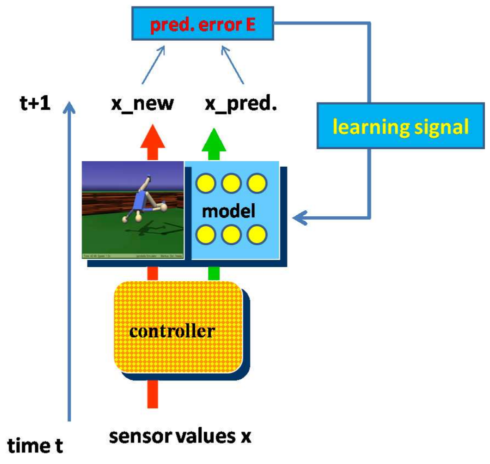


Figure 3: The sensorimotor loop and the self-model of the robot. The controller receives the current sensor values and generates corresponding motor values which are sent both to the robot and its self-model. The difference between the predicted new sensor values and the measured ones forms the prediction error E. A learning signal for the improvement of the model is derived by gradient descending the error E.

2 Self-Regulation – First Ideas

Self-organization and self-determined development are not thinkable without a certain amount of self-awareness of the robot. In particular, the robot should be able to predict the consequences of its actions in the near future. This can be achieved by equipping the robot with an adaptive internal model. We may restrict ourselves in the present setting to the most simple case where the model is to predict the sensor values measured in the next time step, see Fig. 3. These may depend on the executed actions and on the present (and possibly earlier) sensor values.

2.1 The Model

Technically, the model may be understood as a parameterized function $M : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ mapping the current sensor-action state (x_t, y_t) to the next sensor state x_{t+1} such that

$$x_{t+1} = M(x_t, y_t) + \xi_{t+1} \tag{4}$$

where ξ_{t+1} is the misfit between the predicted and true sensor values and

$$E = \xi^\top \xi \tag{5}$$

is the error of the model.

This kind of model plays an important role in our approach. Let us therefore go a bit deeper into detail here. In order to get an understanding of what such a model is about let us consider the BARREL again. Actually, the BARREL with its internal weights driven by restricted motor forces is quite a complicated physical object, with 8 degrees of freedom (three translational, three for the orientation and two for the internal weights), in general interacting with other physical objects like the surface (which may be structured and/or highly elastic) and other obstacles in an intricate way. The BARREL can only be modeled in full generality by using its full equations of motion in some convenient representation, like the Lagrange formalism of classical mechanics, together with the details of the interactions. Given the relevant information, including the coordinates of the BARREL and the structures of both the ground and the obstacles, these equations can be solved and this is exactly what the simulation software does. However, we neither have this information (we have only the inclination of the two internal axes via the sensor values) nor do we want to rely on them. Self-organization, at least as we know it from nature, means the emergence of dimensionally reduced modes in complex physical systems. The rolling mode on even ground is the best example since it can be described by a largely reduced dynamical system.

The reduced dynamical complexity of these "natural" modes is reflected in the complexity requirements for the model. Quite surprisingly we found in numerous applications that a linear or pseudo-linear model is all we need in the pure self-organization phase. In detail, we use, instead of Eq. (4) in the most simple setting

$$x_{t+1} = Ay_t + b + \xi_{t+1} \tag{6}$$

where ξ gives the model error, A is a matrix, and b a bias vector, both of which are subject to a fast and never ending learning dynamics which is the essential compensation for the simplicity of the models.

In an online learning scenario we obtain in each time step a pair $(x_{t+1}; y_t)$ and try to adapt the parameters such that the modeling error is reduced. A typical approach is the gradient descent on the error landscape, see Chapter **??** for details. The update ΔA_{ij} is obtained componentwise as[1]

$$\Delta A_{ij} = \varepsilon_A \xi_i y_j - \lambda A_{ij} \tag{7}$$

$$\Delta b_i = \varepsilon_A \xi_i - \lambda b_i \tag{8}$$

with a conveniently chosen learning rate $\varepsilon_A > 0$ and the decay constant $\lambda \ll 1$ such that the model has a chance to forget outdated contents. The index i is running over all sensors, i. e. $i = 1, 2, \ldots, n$ and j over the motors as $j = 1, 2, \ldots, m$. The update rule, essentially the famous Delta rule (Wikipedia), has a simple interpretation. For instance, ΔA_{ij} is the product of the input x_j into the synapse A_{ij} times the error activity ξ_i at the output of the unit which is reminiscent of Hebb's rule of learning in the brain.

One important point is the choice of time scales. In our simple BARREL case the time scale for the parameter dynamics Eq. (7) is just a few seconds real time by choosing ε_A appropriately, see Fig. 4. This unusual fast learning is possible not only for the pure rolling modes, as we will see later. Learning in this sense does not mean that the parameters converge so that the prediction error is going to zero. Instead it turned out to be sufficient for the self-organization scenario described in the following that relative information is provided in order to discriminate the good (in the sense of natural modes that can be modeled well) from the less interesting regions of the behavior space.

This is good news which, however, leaves the essential question open. If we want such modes to self-organize, how can a model help in this emergence process if it actually "understands" only the product and not its emergence, i. e. how to go there. The controller on the other hand urgently needs the help of the model in order to find its specific control mode. This is a central problem of self-organization, namely that controller learning and model learning have to bootstrap each other.

[1]For clarity the time indices are omitted.

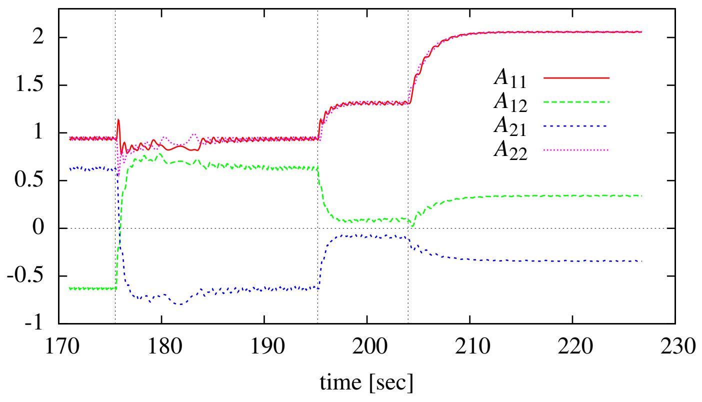


Figure 4: The matrix elements of the model matrix A in a relearning process. The parameters of the controller are modified three times (**dashed vertical lines**). The model is seen to adapt in a very short time to the new situation. In the final situation we had $C_{11} = C_{22} = 0.5$ and $C_{12} = -C_{21} = 0.1$. The relearning times are largely determined by the inertia of the BARREL since the direction of rolling has to change as a consequence of the parameter switching.

2.2 Learning the Controller from Specialist Models

We have seen above, that, given a controller, the model can learn readily the correlations between the actions and the induced sensor values. This is not a surprise given that we accept larger modeling errors and reduced competencies of the model (specialist model). In order to understand more about the mutual bootstrapping of model and controller, we have to consider the inverse problem. Let us assume we have learned a model for a given behavior generated by the parameters C and b of the controller. Can we reobtain the behavior from knowing the model? In principle this should be possible, since the model error can not only be minimized by changing the model for a given behavior, but the error of a given model depends just as well on the behavior. Formally, this becomes obvious when considering the error in Eqs. (4, 6) which depends via the controller output y on the parameters of the controller. However, will this also be possible for the specialist models which are feasible only for the established behavior? The question can not be answered in general but the practical experiences show that such models are well able to attract the controller towards the pertinent behavior.

Learning the controller from the prediction error of the model is schematized in Fig. 5. In the specific setting used above we find, using (3), the learning rules as

$$\begin{aligned} \Delta C_{ij} &= \varepsilon_C \eta_i x_j - \lambda C_{ij}\,, \\ \Delta h_i &= \varepsilon_C \eta_i - \lambda h_i\,, \end{aligned} \tag{9}$$

where

$$\eta_i = \sum_j A_{ji} g'(z_j)\,\xi_i\,, \tag{10}$$

with $z_j = \sum_k C_{jk} x_k + h_j$ and $g'(z) = \tanh'(z) = 1 - \tanh^2(z)$ is the result of back propagating ξ through the output function of the controller. Details may be found in Chapter **??**. Similar to the model case we added a small decay term with $\lambda \ll 1$ to the

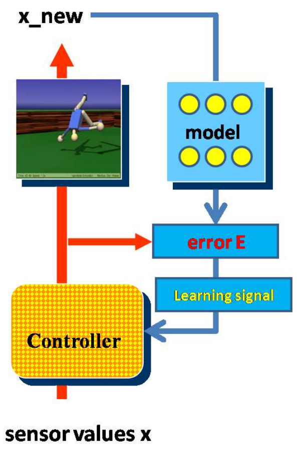


Figure 5: Learning the controller from the model. The task is now inverted. Given the model, how can the controller be adapted so that the behavior of the robot can be well predicted by the model.

learning rule. The matrix A and the column vector b define the model and are assumed to be known for the moment.

Experiment 2.1: Learning controller from self-model

Start the OUTLINE-DEMONSTRATOR and select “Learning controller from model”.
The simulation is now running with ε_C=`epsC=0`, ε_A=`epsA=0.1`. Check the convergence of the matrix A using the GUILOGGER (<Ctrl>+G). After convergence stop learning of the model by entering
```
>epsA=0
```
Change the behavior by pressing ‘s’ or ‘S’ (in the graphical window) multiple times which scatters of the parameters of C by random values in $[-0.2, 0.2]$ or $[-0.5, 0.5]$ respectively.
Switch on learning of the controller:
```
>epsC=0.1
```
Watch whether the previous behavior is reestablished. Try more drastic changes by pressing ‘r’ or ‘R’ to re-initialize C randomly (see Experiment 1.2). Finally you can change start anew by switching to exclusive model learning using `epsC=0`, `epsA=0.1`.

The idea can be corroborated experimentally, see Experiment 2.1 or Fig. 6 which clearly demonstrate that the controller learns in just a few seconds the new behavior prescribed by the model in the case of the pure rolling modes. On the other hand, as the reader easily convinces himself, see Experiment 2.1, learning the controller from a model of the body is also feasible in more complex modes of behavior. For instance, we may use one of the above learned models for the “lolloping” modes. As the experiments show, the controller can be recovered from such a model in nearly the same reliable way as in the rotation mode. This puts some credibility into the postulate that behavior, in the natural modes at least, can be learned from having a **simple** self-model. In other words, these extremely simple models can act as a kind of role model for quite complicated behaviors.

Note that even though the models may have a non-vanishing residual error, the bootstrapping does not fail because mainly the relative errors matter.

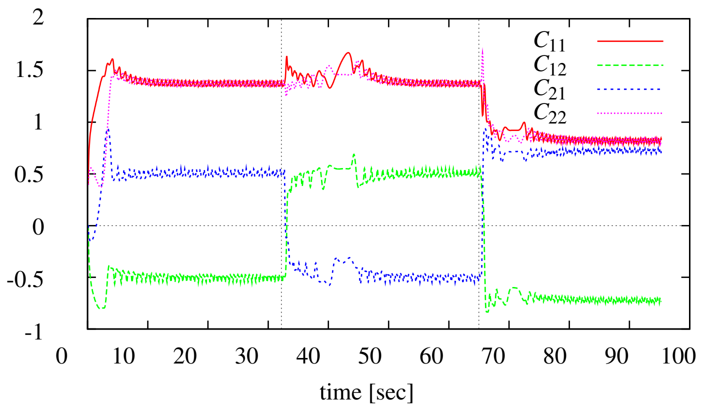


Figure 6: Behavior of the controller matrix C over time when learning the behavior from pre-specified models (rotation matrices). At seconds 32 and 65 the model was externally changed. Between the switches the controller matrix C clearly has the rotation structure. The relearning of the behavior takes only a few seconds.

2.3 Homeostasis: Self-Regulated Stability

We have seen that simple controllers can command complicated behaviors if they excite nontrivial physical objects in the closed loop control mode. On the other hand, we have seen that models with restricted competencies can host very complicated behaviors like rocking, jumping or kind of lolloping in our BARREL case. Moreover, we have seen that these two units can teach each other very effectively. We can imagine this as a kind of mutual, body mediated attraction between these two units. However, this mutual attraction phenomenon was possible only if one of the two was pre-specified from outside or learned in an earlier episode to realize a specific mode of behavior. What will happen if we let the two teach each other simultaneously? Obviously the attraction will now drive both model and controller towards each other. Without any goal or purpose given from outside there are more questions than answers. Will the system go into a specific mode of behavior and which one will it be? Can we expect any systematics from such a procedure and what will the role of the embodiment be in such a process?

The general setting is sketched in Fig. 7. In the above realization for both the model and the controller we simply have to run the Eqs. (7, 9) concomitantly so that we now have a combined dynamics consisting of the BARREL, the model parameters, and the controller parameters, altogether a space of 20 dimensions with comparable time scales.

The idea can easily be checked experimentally, see Experiment 2.2. Considering many different initializations for the C matrix reveals that the concomitant learning of controller and model most of the time runs into a fixed point of the combined dynamics where the robot is at rest, (see Fig. 8. Actually, this is not surprising since in a physically stable situation the model can easily learn the motor-sensor correlations so that the prediction error has a minimum.

On the other hand, when starting the controller or the model with a rotation matrix or if the robot is kicked from outside into a rolling motion, something interesting happens: The BARREL starts rolling, but it will not end up in a fixed rotation mode.

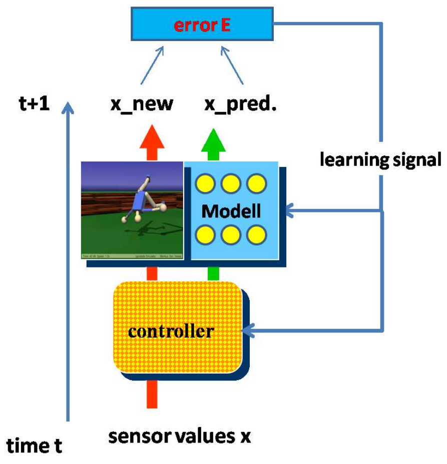


Figure 7: How the controller and the model teach each other. The prediction error provides a learning signal for both the model and the controller.

Experiment 2.2: Homeostatic control of the BARREL

Start the OUTLINE-DEMONSTRATOR and select "Homeostatic control". The parameters are ε_C=`epsC=0.1`, ε_A=`epsA=0.1`. The robot is at rest and will not leave this state by itself. Exert external forces to the BARREL by pressing 'x' or 'X' (in the graphical window) or use <Ctrl>+<left Mouse button> or <Ctrl>+<right Mouse button>.You can speed up/slow down the simulation by pressing '+'/'-'. Surprisingly the robot does not always come to rest but enters a periodic forward/backward locomotion behavior. Try different learning rates when the robot is in such a mode for instance:

```
>epsA=0.005
```

The interval of forward and backward motion becomes longer.

Try `>epsC=0.5` and `>epsA=0.5`. and observe the behavior. Eventually the robot will stabilizes into the "do nothing" regime, since the model can learn the behavior quick enough.

You may re-initialize both the parameters of the controller and the self-model randomly as described in Experiment 1.2. For large learning rates the system come to rest in almost any case. For low learning rates we observe often oscillatory behavior.

Instead it increases velocity then decelerates, reverses velocity and starts the same in the opposite direction. This slowly oscillating mode will then continue forever. So, instead of realizing just one behavior (rolling mode with fixed velocity) the robot sweeps through the behavior space of the possible rolling modes. This is not only seen at the level of behaviors but is clearly reflected by the structure of both the C and A matrices which sweep (more or less periodically) through the space of rotation matrices.

The reason for this behavior is seen in the mutual teaching scenario of model and controller. Both components are trained to imitate each other but since the learning needs time one of the two is always lagging behind the other with a change of roles after some time. Most importantly the model is lagging behind the physical mode. This is also supported by the experiment: for large learning rates the system comes to rest most of the time – the lag between model, controller and physical mode is small. However if the learning rates are small this lag is significant and we observe the periodic forward/backward behavior. In Chapter **??** we will trace back this behavior to a very general phenomenon, the spontaneous symmetry breaking which is ubiquitous in self-organizing systems.

The preceding investigations show that, depending on the starting conditions and/or external perturbations, the mutual attraction mechanism between model and controller provides a self-regulation into body related behaviors which may be oscillatory or at a stable fixed point attractor. The latter correspond to stable physical states where the body is at rest. The system can be switched from one to another by strong external mechanical perturbations, see Experiment 2.2.

Nevertheless, so far there is no intrinsic drive for innovations that might lead the system out of these states. The most obvious reason for the lack of initiative is seen in the fact that physically stable situations are candidates for small prediction errors, since the models can most easily adapt to such a situation. In exceptional cases, this may produce unexpected results, like the sweeping mode, if we have an exceptionally high degree of symmetry in the system. In other applications, like the humanoid robot, one finds that the landscape of behaviors is mainly characterized by stable attractors of the dynamics. In a sense, the method described so far realizes a version of Ashby's homeostasis [1] or even ultra-stability with self-determined set points.

3 Homeokinesis: Body Inspired Control

The principle found so far is appealing because it brings the embodiment of the robot into play. In fact, since we do not give any goals or external cues, it is the embodiment that decides where the system tends to develop. In the experiment the controller drives the system into physical states which it can stabilize best. However, in this way one does not get the curious and self-exploratory robot we want to create. Instead of realizing a general stasis in the system we want brain, body, and environment to get into a common kinetic regime. So, instead of homeostasis we want a *homeokinetic* system.

The central question is now: How can we find a general drive for activity? Or phrased differently: How can we get out of the stable attractors? This would be simple if we could invert the arrow of time. Of course, the physics of the system can surely not run backward in time. However, what we can do is to let the controller learn to stabilize the behavior backward instead of forward in time as before. This means we must consider *postdiction* instead of prediction and reiterate the previous considerations. Postdiction means that the earlier sensor values are reconstructed from the current ones, by means of a model. The sequence of steps from x_t trough the world to x_{t+1} and

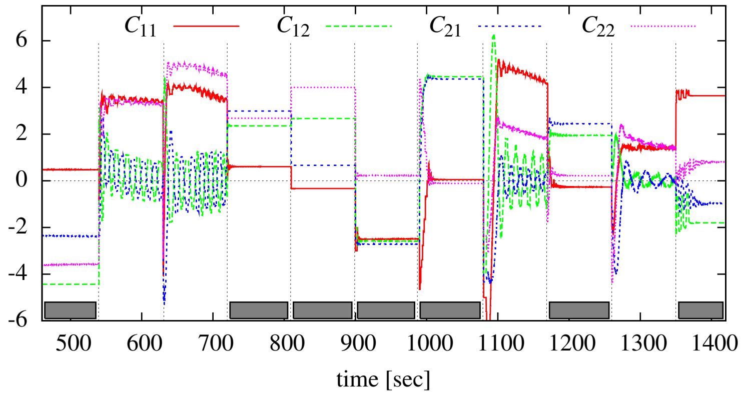


Figure 8: A long run of the BARREL with random re-initialization of the C matrix every 90 sec. The re-initialization of the controller matrix is followed by a relearning of both the model and the controller. In most cases (marked by gray bars) there is a rapid convergence towards a stable situation. In other cases the learning dynamics remains active which corresponds to a richer dynamics of the system in the sweeping modes. Parameters: $\varepsilon_C = 0.5, \varepsilon_A = 0.1$, no friction.

back through the model and the controller to $x_{reconstr.}$ forms a time loop so that the postdiction error is also called the time loop error. The general scheme is depicted in Fig. 9 and a theoretical foundation will be given in Chapter **??**.

What we expect is that oscillatory patterns like the ones observed in the frequency sweeping effect above, stay the same since (quasi-)periodic motions look the same forward and backward in time. Otherwise we expect a weak destabilization of the system and this is what we need for the self-organization, see Chapter **??**.

Gradient descending the postdiction error instead of the prediction error leads to new update rules for the parameters of the controller, see Chapter **??**. Using our previous setting, we give the rules explicitly in order to get some first impression of their structure and function. We introduce the column vectors[2] $\zeta = (AG')^{-1}\xi$, $\mu = \left(CC^T\right)^{-1}\zeta$, and $v = C\mu$, as auxiliary quantities, the squashing function reducing numerical instabilities. The time loop error is

$$E_{TLE} = v^T v = \zeta^T \frac{1}{CC^T}\zeta\,. \tag{11}$$

With the learning of the self-model given as before by Eq. (7) the gradient descent on E_{TLE} yields

$$\begin{aligned}\Delta C_{ij} &= \varepsilon_C \mu_i v_j - 2\varepsilon_i y_i x_j - \lambda C_{ij}\,,\\ \Delta h_i &= -2\varepsilon_i y_i\end{aligned} \tag{12}$$

which replaces Eq. (9). Indices are running over all sensors and motors as before, i. e. $i = 1,2,\ldots,n$ and $j = 1,2,\ldots,m$ and we introduced the channel specific learning rate

$$\varepsilon_i = \varepsilon_C \mu_i \zeta_i\,.$$

[2]The matrix inverses have to be understood as pseudoinverses if the normal inverse does not exist.

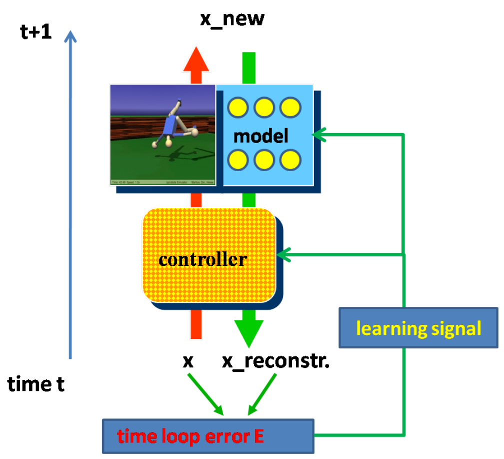


Figure 9: The time loop error. The new sensor values are now backpropagated through both the self-model and the controller to yield the reconstructed previous sensor values. The difference between reconstructed and true sensor values forms the postdiction or time loop error.

These relatively simple update rules define the parameter dynamics of the controller, the learning of the self-model being given by Eqs. (7, 8). The rules need some numerical precautions which are presented in Chapter **??**. As before, learning is not to be understood as the convergence towards a specific goal. Instead, the learning rates usually are chosen such that the parameter and system dynamics run on comparable time scales. In the neural network interpretation we have a fast synaptic dynamics which is constitutive for the behavior of the system. Before going to discuss these rules in more detail, it will be helpful to demonstrate their function when applied to the BARREL.

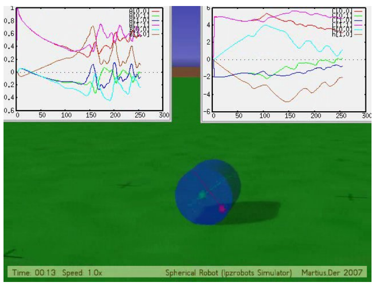
3.1 The BARREL Case

The BARREL, as introduced in Section 1, has two motors shifting the internal weights and two sensors measuring the inclination of the internal axes. Hence we put $m = n = 2$ in the above Eqs. (7, 12) and define the learning rates empirically to make learning sufficiently fast. When connecting the homeokinetic "brain" to the robot the controller can be initialized so that it stabilizes the robot in a resting position. Then, if the learning is switched on, the robot starts to roll after a short time in one direction and in most cases enters the sweeping mode which we already observed in the homeostatic setting (Section 2.3). This destabilization effect, clearly demonstrated in Video 3.1, is the novel and important property which makes the system much richer in the spectrum of behaviors. In the experiments one observes a lot of further active modes besides the sweeping mode. Many of them are long lived transients but sometimes changing between modes must be helped by either re-initializing the parameters or by shaking the robot by external force. Through mechanical influences one may provoke a drastic change of parameters, as one can observe in the parameter inlets of Video 3.1 or by doing Experiment 3.1. It is interesting to see how the system reorganizes in a few seconds or minutes into a new active, but seemingly organized mode of behavior.

The simultaneous learning of both model and controller from the time loop error is

Experiment 3.1: Homeokinetic control of Barrel

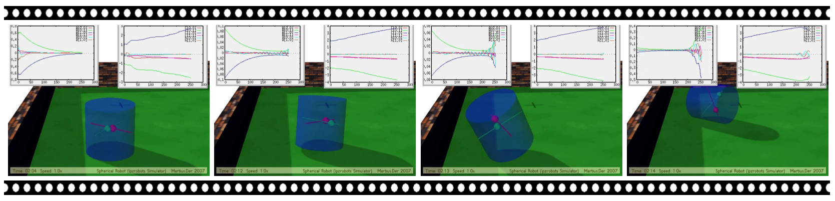
Start the Outline-Demonstrator and select “Homeokinetic control”. The robot is at rest. Observe the generated behavior and the parameter dynamics with the GuiLogger. Remember that you can change the simulation speed with ‘+’/‘-’. Try different learning rates for instance by entering > epsC=value
You may re-initialize both the parameters of the controller and the self-model randomly as described in Experiment 1.2.
In order to exert external forces to the barrel use either the <Ctrl>+<left Mouse button> or the <Ctrl>+<right Mouse button>.
In order to make the experiment with the barrel in the upright position close the simulation and select “Homeokinetic control (upright)”.



Video 3.1: Behavior of the Barrel when starting in a situation where the center of gravity of the Barrel is very low so that the situation is physically stable. The homeokinetic learning is seen to destabilize the system quite rapidly (a few seconds real time) and starts to move the Barrel. The dynamics of the parameters of the model and the controller can be followed in the panels at the left and right upper corners, respectively. The panels depict the course of the parameters in a time window of 250 steps corresponding to about 10 sec. After some time we obtain the stable rolling patterns. Later on (at time 01 : 05) the Barrel was stopped by applying a physical force to it (note the red dot which pulls the robot). After releasing the force, the system recovers again in a very short time and resumes the rolling patterns in a slightly modified form. The videos can be found at `http://playfulmachines.com`.

also the source of creativity in handling unexpected situations. The BARREL has been seen to produce interesting rolling and other modes, which however more or less rely on the fact that getting a barrel to roll is quite "cheap" since this is the most natural motion when lying on a surface. What if the BARREL is taken and put upright on the table? This is a completely new situation which does not fit very well into the picture. The problem is, that the internal axes are now horizontal so that the sensor values are equal to zero (apart from some small noise) and thus there is no reaction of the sensors to the motor actions. Nevertheless, if the approach disposes of some intrinsic creativity it should find a way out towards new activity. Video 3.2 shows one possible development in such a situation. After some time the motors start a kind of jiggling motion which is essentially largely amplified sensor noise. But as soon as the BARREL starts to give a definite reaction, the learning system synchronizes with it and amplifies the swaying motion of the robot. As a consequence, the BARREL is falling over and now is free to go into a rolling mode again.

This is not yet the full story. One can, out of this impasse situation, also observe the emergence of a new mode, a kind of precession of the BARREL which requires a high degree of sensorimotor coordination, as presented in Video 3.3. Nevertheless, the mode is quite stable and can last over a long time, about five minutes real time in this particular case. This mode is a truly emerging phenomenon that is produced through the interaction of the learning dynamics with the specific embodiment in the given physical situation. Note again, that both controller and self-model receive nothing but the inclinations of the internal axes so that the physical state of the body remains widely obscure to the "brain". In particular, there is no information about being put upright.



Video 3.2: Creativity in unexpected situations: The BARREL was put into an upright position by an external force about 10 seconds ago. Internal axes are horizontal now so that the sensor values, the inclination of the internal axes, are zero apart from some small sensor noise. The self-model does not get any reliable information in this situation so that rapid forgetting sets in. This is counteracted by the controller which increases weights so that any small perturbations are amplified. After some time the motion of the internal weights become so strong that the BARREL is tossed over. The insets are the same as in Video 3.1. Note that the scales of the panels change, in particular the model parameters change by two orders of magnitude. The rapidly oscillating parameters are the bias terms of both model and controller. The videos can be found at `http://playfulmachines.com`.

3.2 Principles of Action

The emergence of quite unexpected but body related modes in systems with completely unknown physical properties is a general phenomenon of the homeokinetic learning rules. This has been proven in applications with a great variety of machines ranging in complexity from the BARREL to snake and humanoid like artifacts with more than 20 independent degrees of freedom. Nothing of the specific properties of these systems is made explicit to these rules. Instead the latter are used with only minor modifications for any application. The results are even more surprising given the fact that the controller is extremely simple and, without learning, has no internal dynamics of its own.

The peculiarities of the general learning rules will be discussed in depth in Chapter **??** but we will give already here some essential points that can easily be understood from the above formulas. The essential new feature is that the landscape of the time loop error, Eq. (11), is characterized by singularities acting as repeller (infinitely repulsive regions) for the gradient flow. This is a direct result of learning the system backward in time. These repulsive regions are easily identified. A first class of singularities is given by the zeros of g' (featuring in ζ), i. e. in the saturation regions of the neurons. The effect of this singularity, which leads to the anti-Hebbian term $-2\varepsilon_i y_i x_j$ in Eq. (12), is to keep the neurons away from the saturation regime. This is very reasonable since in that region neurons are not sensitive to their inputs.

Further repulsive regions result from the inverted matrix CC^T [3] leading to the first term in the learning rules for both C and h. The role of this singularity is most immediately seen if the robot is initialized in the *tabula rasa* situation ($C = 0$, $h = 0$) so that the controller does not react to its sensor values. The slightest perturbation will quickly drive the parameters of the controller away from this instable fixed point of the combined dynamics so that feedback in the sensorimotor loop is generated and the robot is quickly driven away from this “do nothing” region.

Another singularity is produced by the inverse of the self-model matrix A that appears in ζ. This makes the time loop error very large if the self-model is close to the “know nothing” state. Then, high learning activity of the controller is generated producing a kind of flurry in the system, which produces new behaviors and hence new learning signals for the self-model so that the A matrix will soon leave that region. This is best seen in the Videos 3.2 and 3.5 when the A matrix is close to zero. Thus, there is a circular dependency, A is driven indirectly by the change of behavior (it has caused itself) generated by the change of the controller, that in turn is modulated by A. Importantly, the ultimate source of this interplay is the modeling error ξ which reflects the reactions of the robot to the controller signals. Metaphorically speaking, the gradient flow is directed away from the singularities marking inactivity, lack of knowledge, and of sensitivity (due to saturation) in a way which is determined by the mutual attraction between model and controller which in turn is mediated by the embodiment.

Behaviors generated in this way are inherently contingent (there is no influence from outside) but by far not arbitrary since the whole bootstrapping process is driven by the specific reactions of the embodied robot to the controller signals. Thus, it is the body itself which plays the most active part in the emerging control process so that we have called this approach body inspired control.

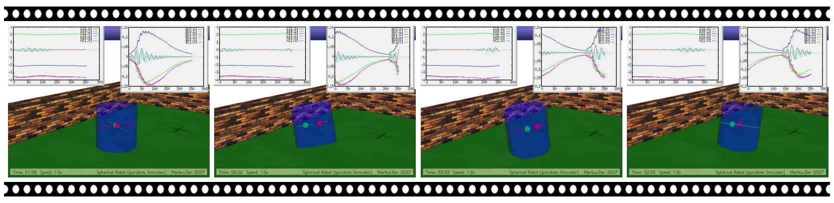
In practical applications we observe highly complicated motion patterns emerging from this method of self-organization. This may seem as a contradiction given the low internal complexity (seemingly oversimplified self-model and controller) of the robot. There are two reasons to resolve this contradiction. One is that in the closed loop control paradigm, it is possible to sustain a complex mode like the precession of the barrel by such a simple controller: Most of the complexity is provided by the body, quite in the sense of morphological computation. The other one is the fact that we do have the fast parameter dynamics driven by the learning rules and it is this simultaneity

[3] In order to avoid numerical problems we do have to introduce some regularization, see Chapter **??**.

of the dynamics that is responsible for the emergence of and transition between modes. Moreover, the error landscape changes all the time due to the simultaneity of parameter and system dynamics, which is the reason why most of the behaviors are transient.

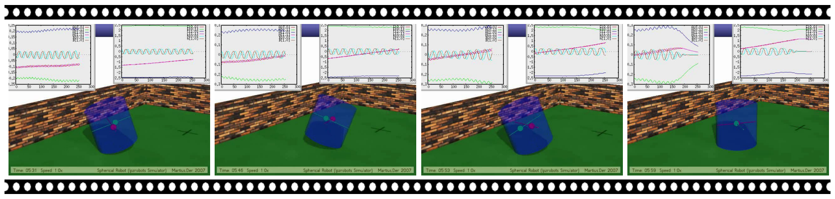
The transient nature of emerging behaviors is nicely demonstrated by the BARREL experiments. For instance, Video 3.5 shows the decay of a precession mode after existing several minutes. The behavior of the parameters of both the controller and the self-model strongly suggests a systematic tendency towards the decay of the mode. This phenomenon is also observed in higher dimensional systems and is seen as a decisive advantage over the common dynamical system approaches which usually try to associate behaviors (or cognitive states) with stable attractors, instead of transients, where no natural way exists to get out of a behavior.

The presented method might at the first glance appear to be a very complex and fragile scheme but our experiences in a large number of applications in highly dimensional and complex systems have shown that this is indeed a reliable scenario. The simplicity of both model and controller play an important role in keeping the pieces together. Realized in this way the time loop error is a feasible objective for a systematic theory of self-organizing behavior systems.

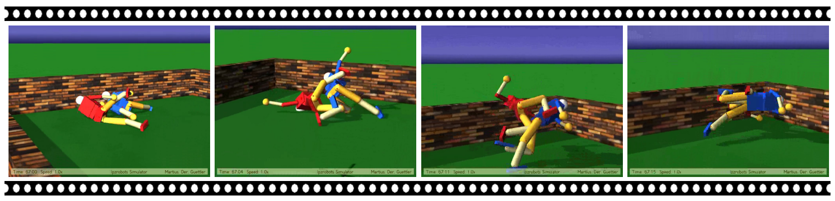


Video 3.3: Emergence of nontrivial modes. Out of the upright position there are different modes that can emerge. In the video you see the emergence of a precession mode which lasts for more than five minutes. Parameter and model dynamics is depicted in the panels in the right and left upper corners (interchanged), respectively. Note that the scales of the panels change, in particular the model parameters change by two orders of magnitude. The videos can be found at `http://playfulmachines.com`.

Video 3.4: The emergence of nontrivial modes like in the barrel case are a common phenomenon when using the general learning rules, cf. Eq. (12) for different agents. In the video you see a simulated humanoid robot. The joints are actuated by simulated servomotors. Sensor values are the measured joint angles. There is no other information available to the robot. After the robot has fallen into a narrow pit, it develops different new motion patterns adapted to the new situation. After some time one often observes the emergence of climbing like behavior patterns. The patterns, although they might help to get the robot out of this impasse, emerge without any goals as a consequence of the sensitive but active interplay between robot and the specific environment.



Video 3.5: Decay of nontrivial modes. One special feature of our approach is that modes do not last forever, instead the concomitant learning of model and controller most often leads to a slow change of the parameters so that the system leaves the mode. In the present case, the precession mode, after lasting for about five minutes decays spontaneously. The parameter change is most prominent in the diagonal elements of both the model (left) and the controller (right panel) depicted by the red and purple lines (which almost coincide).



Video 3.6: Emerging motion patterns are transient and depend decisively on the interaction with the environment. Most complex patterns may emerge if the "environment" is another robot of the same kind. The only sensors are the measured joint angles, like in Video 3.4 so that the two humanoids can only "feel" each other by the mismatch between true (measured) and nominal joint angles resulting from the load on the joint. Adherence is due to normal friction but essentially also from a failure of the ODE physics engine, which after heavy collisions produces an unrealistic penetration effect which acts like a special gripping mechanism. So, the "fighting" is a truly emerging phenomenon not expected before we saw it.

3.3 Discussion

We have given an approach to the self-organization of autonomous robots which is systematic since it reduces the self-organization to a self-supervised learning process, defined by the gradient descent on a single quantity, the time loop error. The gradient flow is made explicit in terms of simple learning rules which can be applied with only minor modifications to a wide variety of physical systems. The approach has a number of unusual features. Unusual is the very simple controller structure, one layer feed-forward neural networks, which seem to be much inferior to the recurrent neural networks used widely in the community. However, this is sufficient under a closed loop paradigm where complexity can be generated by just exciting the dynamics of the body in a convenient way. Additionally, there is a second order recurrence by the fast synaptic dynamics as driven by the gradient flow. In this way, the emerging behaviors, besides being contingent, become also transient, so that behavior architectures may emerge. From the point of view of system theory we have a new class of dynamical systems, called self-referential dynamical systems, which are qualified by the fact that they define their parameter dynamics exclusively on the basis of their own system dynamics. Some fundamental properties of such systems, in particular the spontaneous symmetry breaking scenarios, are analyzed in some generality in this book.

Unusual is also the attitude towards what a robot is to do. In robotics, one is used to have a specific task which is to be solved effectively. Embodied AI opens new perspectives in that the embodiment is intended to help in the process. Homeokinesis gives the robot much more autonomy in that, besides of minimizing its time loop error, no tasks or goals are formulated. This produces activities which are completely free of any purpose but, since they are body inspired, they are the most natural modes for a given body. We have demonstrated in numerous examples how this paradigm makes the robots to unfold their bodily affordances in a playful way.

This body inspired control opens new perspectives in several directions. In particular, for developmental robotics it may be immediately helpful for the realization of the early sensorimotor stage in Piaget's scheme. From a more practical point of view the emerging patterns may be used for the building blocks of a more elaborate behavior architecture. This will be worked out in Part II which is devoted to a new research direction, the so called guided self-organization.

References

[1] W. R. Ashby. *Design for a Brain.* Chapman and Hill, London, 1954.

[2] S. H. Collins and A. Ruina. A bipedal walking robot with efficient and human-like gait. In *IEEE Conf. on Robotics and Automation, ICRA 2005*, pages 1983–1988. IEEE Press, 2006.

[3] R. Der. Self-organized acquisition of situated behavior. *Theory in Biosciences*, 120:179–187, 2001.

[4] R. Der, F. Hesse, and G. Martius. Learning to feel the physics of a body. In *Proc. Intl. Conf. on Computational Intelligence for Modelling, Control and Automation (CIMCA 06)*, pages 252–257, Washington, DC, USA, 2005. IEEE Computer Society.

[5] R. Der, F. Hesse, and G. Martius. Rocking stamper and jumping snake from a dynamical system approach to artificial life. *Adaptive Behavior*, 14(2):105–115, 2006.

[6] R. Der and R. Liebscher. True autonomy from self-organized adaptivity. In *Proc. of EPSRC/BBSRC Intl. Workshop on Biologically Inspired Robotics*, HP Labs Bristol, 2002.

[7] R. Der and G. Martius. From motor babbling to purposive actions: Emerging self-exploration in a dynamical systems approach to early robot development. In S. Nolfi, G. Baldassarre, R. Calabretta, J. C. T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, and D. Parisi, editors, *Proc. From Animals to Animats 9 (SAB 2006)*, volume 4095 of *LNCS*, pages 406–421. Springer, 2006.

[8] R. Der, G. Martius, and F. Hesse. Let it roll – emerging sensorimotor coordination in a spherical robot. In L. M. Rocha, L. S. Yaeger, M. A. Bedau, D. Floreano, R. L. Goldstone, and A. Vespignani, editors, *Proc, Artificial Life X*, pages 192–198. Intl. Society for Artificial Life, MIT Press, August 2006.

[9] R. Der, U. Steinmetz, and F. Pasemann. Homeokinesis - a new principle to back up evolution with learning. In *Proc. Intl. Conf. on Computational Intelligence for Modelling, Control and Automation (CIMCA 99)*, volume 55 of *Concurrent Systems Engineering Series*, pages 43–47, Amsterdam, 1999. IOS Press.

[10] G. Martius and R. Der. Simulation software for the this book. `http://playfulmachines.com/book/software`, 2010.

[11] G. Martius and J. Herrmann. Taming the beast: Guided self-organization of behavior in autonomous robots. In S. Doncieux, B. Girard, A. Guillot, J. Hallam, J.-A. Meyer, and J.-B. Mouret, editors, *From Animals to Animats 11*, volume 6226 of *LNCS*, pages 50–61. Springer, 2010.

[12] G. Martius, J. M. Herrmann, and R. Der. Guided self-organisation for autonomous robot development. In A. e Costa and Francesco, editors, *Proc. Advances in Artificial Life, 9th European Conf. (ECAL 2007)*, volume 4648 of *LNCS*, pages 766–775. Springer, 2007.

[13] T. McGeer. Passive dynamic walking. *Int. Journal of Robotics Research*, 9(2):62–82, 1990.

[14] R. Smith. Open Dynamics Engine – open source, high performance library for simulating rigid body dynamics. `http://ode.org`, 2008.