Self-Adjusting Ring Modules (SARMs) for Flexible Gait Pattern Generation

Manfred Hild[†], **Frank Pasemann**[‡]
† Institut für Informatik, Humboldt-Universität zu Berlin, hild@informatik.hu-berlin.de
‡ Fraunhofer Institut AIS, Sankt Augustin, Germany, frank.pasemann@ais.fraunhofer.de

Abstract

Using the principle of homeostasis, we derive a learning rule for a specific recurrent neural network structure, the so-called Self-Adjusting Ring Module (SARM). Several of these Ring Modules can be plugged together to drive segmented artificial organisms, for example centipede-like robots. Controlling robots of variable morphologies by SARMs has major advantages over using Central Pattern Generators (CPGs). SARMs are able to immediately reconfigure themselves after reassembly of the robot's morphology. In addition, there is no need to decide on a singular place for the robot's control processor, since SARMs represent inherently distributed control structures.

1 Introduction

In order for an organism to move, it is necessary that its limbs and its body itself move in a coordinated manner. A special case of coordination is found in segmented organisms like centipedes, where similar parts of the body move in the same way, but phase-shifted. If we want to implement an artificial organism that moves, i.e. an autonomous robot, we can use a central pattern generator (CPG) to drive the motor joints and thus select one of the well documented architectures from the literature [Golubitsky *et al.*, 1998]. But when a modular structure that can be assembled during runtime is asked for, it is unclear where a CPG should be located, so a distributed approach will better fit the requirements.

In the following sections we introduce a concept of distributed control for the generation of locomotion, based on a single neural building block, the Self-Adjusting Ring Module (SARM).

2 Two Neurons Form an SO(2)-Oscillator

Using a discrete time, fully connected recurrent neural network with two neurons, no bias terms, and $tanh$ as transfer function, the only parameters to specify are the weights which form a 2×2-matrix W_2. This matrix is based on a rotation matrix [Thompson and Steward, 2002], i.e an element of the unitary group SO(2) and has the form

$$W_2 = (1+\epsilon)\begin{pmatrix} s & r \\ -r & s \end{pmatrix},$$

where

$$s = \cos(2\pi\phi),$$
$$r = \sin(2\pi\phi),$$

and

$$0 \leq \phi \leq 1/2,$$
$$0 < \epsilon.$$

Here s stands for the neurons' recurrent self-connections, and r for the ring-connections between the neurons. If we choose $\epsilon \ll 1$ and thus compensate for the damping property of the transfer function $tanh$, the two neurons approximately oscillate with frequency $f_0 = \phi f_s$, relative to the update or sampling frequency f_s of the network [Pasemann *et al.*, 2003]. Since the two neurons produce almost sine waves that differ in phase by $\pi/2$, it is possible to generate a sine wave and other waveforms of the same frequency, but with arbitrary amplitude and phase, just using a weighted sum of the two neurons' output signals [Hornik *et al.*, 1989; Cybenko, 1989].

2.1 A Simple Ring Module

We can extend the SO(2)-Oscillator to the more general form of a ring oscillator by introducing more neurons [Pasemann, 1995]. Using m neurons ($m \geq 2$), the weights can be written as the following $m\times m$-matrix

$$V_m = (w_{i,j}) = \begin{pmatrix} s & r & & \\ & \ddots & \ddots & \\ & & s & r \\ -r & & & s \end{pmatrix},$$

where again s stands for the self-connections, r for the ring-connections, and all other entries are zero. For the ring to oscillate we choose the condition

$$\left(\prod_{i=2\ldots m} w_{i-1,i}\right) w_{m,1} < 0,$$

which for $n \geq 1$ and $m = 2n$ is fulfilled by the following alternative arrangement of weights:

$$U_{2n} = \begin{pmatrix} s & r & & & \\ & s & -r & & \\ & & \ddots & \ddots & \\ & & & s & r \\ \beta r & & & & s \end{pmatrix}, \qquad \beta = (-1)^n.$$

Clearly we have $W_2 = V_2 = U_2$, but for $n > 1$ always $V_{2n} \neq U_{2n}$. If n is odd, then $\beta = -1$, therefore U_{2n} can be implemented as n identical copies of the following ring module:

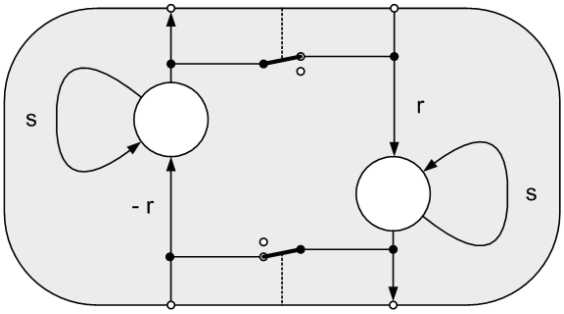

Figure 1: Structure of a Ring Module

The modules have a switch at each end which connects the two outputs as long as no other module is plugged in, thus single modules oscillate stand-alone. When modules are plugged together, their switches open up, so that the former two rings unite to one. For even n one needs to introduce a negation at some position in the ring, e.g. by means of a simple switch.

Interesting applications of those modules emerge, if they have motor joints included, which are driven by a linear combination of the modules' internal neurons. Flipping of one ring module within a chain of modules changes the phase relationships of all modules.

2.2 Varying Frequency and Amplitude

The mapping $s = \cos(2\pi\phi), r = \sin(2\pi\phi)$ no longer holds for the ring oscillators of type U_{2n}, because the additional neurons introduce a delay which in turn considerably lowers the frequency. At the same time the amplitude rises, since there are more self-connections present which boost up the energy of the whole system.

There does not exist a simple formula for calculating s and r as a function of ϕ, but if the desired frequency is relatively low compared to the network's update frequency, i.e. for small ϕ, we find that:

$$s = \cos(2\pi\phi) \approx 1,$$
$$r = \sin(2\pi\phi) \approx 2\pi\phi.$$

In other words, the weights of the self-connections always stay close to 1, mainly controlling the level of energy flow in the system and therefore the amplitude, whereas the weights of the ring-connections are approximately proportional to the frequency. The proportion ratio depends on the number of neurons. This is exactly what we will need to implement a ring module that is able to self-adjust its weights in order to maintain a certain frequency and amplitude.

3 Introducing the Principle of Homeostasis

A central tenet of modern physiology is homeostasis [Der and Pantzer, 1999]. It describes the principle, that every living organism tries to settle down to a healthy internal state after it has been disturbed. This desired state is also called an equilibrium. We define the ring module to have reached its equilibrium, if it oscillates with target frequency f_0 and target amplitude A_0. The phase shift will not be of concern here. In order to apply homeostatic mechanisms to the ring module, we first have to introduce an appropriate measure for the equilibrium [Urzelai and Floreano, 2001].

3.1 Discrete Sampled Sine Waves

If we sample one full period of a sine wave at N discrete time steps we can study the following two sums:

$$S_A(A,N) = \frac{1}{N}\sum_{k=0}^{N-1}\left|A\sin\left(\frac{2\pi k}{N}\right)\right|,$$

$$S_f(A,N) = \frac{1}{N}\sum_{k=0}^{N-1}\left|A\sin\left(\frac{2\pi(k+1)}{N}\right) - A\sin\left(\frac{2\pi k}{N}\right)\right|.$$

The first sum $S_A(A,N)$ calculates the average amplitude of the rectified signal and can easily be approximated by:

$$\begin{aligned} S_A(A,N) &\approx \lim_{N\to\infty} S_A(A,N) \\ &= \frac{1}{2\pi}\int_0^{2\pi} |A\sin x|\,dx \\ &= \frac{2}{\pi}\cdot A. \end{aligned}$$

Thus, a sine wave of peak amplitude A results in an average absolute neural activity of $2A/\pi$, independent of the frequency. For $N > 3$ the second sum $S_f(A,N)$ can be approximated by

$$\begin{aligned} S_f(A,N) &\approx \frac{4}{N}\sum_{k=0}^{\lfloor N/4\rfloor-1}\left|A\sin\left(\frac{2\pi(k+1)}{N}\right) - A\sin\left(\frac{2\pi k}{N}\right)\right| \\ &= \frac{4A}{N}\sin\left(\frac{2\pi\lfloor N/4\rfloor}{N}\right) \\ &\approx \frac{4A}{N}. \end{aligned}$$

The approximation error

$$E_f(A,N) = \frac{4A}{N} - S_f(A,N) \geq 0$$

is zero if N is a multiple of four. In general, the relative error is below 3% for $N > 10$, and below 1% for $N > 18$. Altogether, assuming a constant amplitude, the average variation of neural activity is reciprocally proportional to the oscillation period.

3.2 The Learning Rules

Having defined an equilibrium condition, we now need some example sets for weights of stable oscillating ring networks. As a starting point, we assume a network update frequency of $f_s = 50$Hz, which corresponds to the timing of standard servo motors. Choosing an oscillation frequency of $f_0 = 1$Hz as moderate speed for a walking machine, we have $\phi = f_0/f_s = 1/50$.

The larger the amplitude, the larger the distortions of the sine wave, due to the non-linearity of $tanh$, so using $A_0 = 0.4$ as target amplitude is a good choice. Varying the number of ring modules $n \in \{1, 2, \ldots, 6\}$ with U_{2n} as corresponding weight matrix, we get the following results by numerical simulation:

$2n$	s_{2n}	r_{2n}	$s_{2n} + r_{2n}$
2	1.036	0.132	1.168
4	0.905	0.187	1.092
6	0.814	0.260	1.074
8	0.721	0.345	1.066
10	0.642	0.420	1.062
12	0.550	0.510	1.060

Table 1: Weight Settings

If we start with three coupled ring modules and then add one module, the amplitude grows by $\Delta A = 0.2$ from 0.4 to 0.6. In order to compensate for this effect, we have to adjust the self-coupling weights by $s_8 - s_6 = -0.093$. If we leave out one module, the weights have to be adjusted accordingly by $s_4 - s_6 = +0.091$, so the average absolute adjustment is $\Delta s = 0.092$. Putting this together, we end up with a simple learning rule for the self-coupling weights, which stabilizes the amplitude:

$$s(t+1) = s(t) + s_\delta(t),$$
$$s_\delta(t) = \varepsilon_s \frac{\Delta s f_0}{\Delta A f_s} \left(\frac{2}{\pi} A_0 - |\sigma(t)| \right),$$

where $s(t)$ is the weight and $\sigma(t)$ the neuron's output at time t, A_0 and f_0 are target amplitude and frequency, and Δs, ΔA, and f_s are defined as above. ε_s is the normalized learning rate, i.e. $\varepsilon_s = 1.0$ means, that after changing the number of connected ring modules, the target amplitude A_0 will be reached again after the next full oscillation period.

Two things should be noted: firstly, the appearance of f_0 in the learning rule is just part of the learning rate's normalization and not essential for the proper amplitude regulation. Secondly, since all parameters are known in advance, the formula can easily be implemented as

$$s_\delta(t) = \eta \left(\mu - |\sigma(t)| \right),$$

with pre-calculated constants η and μ, according to A_0, ΔA, f_0, f_s, ε_s, and Δs. Thus, only a multiplication, an addition, and a sign flip is needed. Because of the computationally low cost, this learning rule can run even on simplest 8 bit-microprocessors [Hikawa, 2003].

In an analogous manner we get a learning rule for the weights of the ring-connections, which stabilizes the oscillation frequency:

$$r(t+1) = r(t) + r_\delta(t),$$
$$r_\delta(t) = \varepsilon_r \frac{\Delta r}{\Delta S_f f_s} f_0 \left(\frac{4A_0}{f_s} f_0 - |\sigma(t) - \hat{\sigma}(t)| \right),$$

where $r(t)$ is the weight at time t, $\sigma(t)$ and $\hat{\sigma}(t)$ are the outputs of the neurons after and before the r-weighted connection, respective, ε_r is the normalized learning rate, and the rest as defined above. Again, this learning rule can also be written in a simple form as

$$r_\delta(t) = \eta \left(\mu - |\sigma(t) - \hat{\sigma}(t)| \right),$$

with pre-calculated constants η and μ. Note, that despite the term $-\hat{\sigma}(t)$, both formulas are identical. Clearly, $\hat{\sigma}(t)$ must be zero in the first learning rule, since there is only one neuron involved with a self-connection.

Since $r_\delta(t)$ controls the oscillation frequency, but depends on A_0 and f_0 at the same time, this learning rule will only work properly combined with the learning rule for the self-connections. Care has to be taken with the choice of ε_s and ε_r, because too high a value will deform the waveform's shape and cause uncontrolled interactions between the two learning rules. Good parameter choices are given at the end.

3.3 Diffusion Processes

If we assemble some ring modules with well-tuned parameters and then switch them on, the learning rules reliably adjust oscillation frequency and amplitude within a few oscillation periods. This will work equally well for a single module or up to six and more modules.

Now, if we separate some modules and reassemble them all during runtime, something special may happen. Since the separated modules go on oscillating and adjusting their weights independently from each other, we have an unpredictable distribution of weight values and neural activities at the time we reconnect the modules. The modules then end up with an unequal weight distribution. Frequency and amplitude will be well preserved by the learning rules, but the relative phase shift between the modules on one side, and between the two neurons within one module on the other side will no longer be the same. In order to compensate for this, we introduce two independent diffusion processes, one for each weight pair s, s' and r, r' within a module, as follows:

$$\bar{s}(t) = \frac{s(t) + s'(t)}{2},$$
$$s(t+1) = \delta_s \bar{s}(t) + (1 - \delta_s)\, s(t),$$
$$s'(t+1) = \delta_s \bar{s}(t) + (1 - \delta_s)\, s'(t),$$

and similarly for the ring-connections:

$$\bar{r}(t) = \frac{r(t) + r'(t)}{2},$$
$$r(t+1) = \delta_r \bar{r}(t) + (1 - \delta_r)\, r(t),$$
$$r'(t+1) = \delta_r \bar{r}(t) + (1 - \delta_r)\, r'(t).$$

The diffusion processes have to be applied immediately after the learning rule's weight update. Generally it is sufficient to use small values for δ_s and δ_r, i.e. $0 \leq \delta_s, \delta_r \ll 1$.

When implemented on a simple 8 bit-microprocessor, as mentioned above, only additions are needed, since a division by two can be implemented as a right shift. The same is true for smart choices of δ_s and δ_r.

4 Results and Discussion

To illustrate the effect of the homeostatic learning rules and diffusion processes, we simulate hot plugging and unplugging of one to five continuously operating SARMs, first with fixed weights, then with homeostatic control of the weights. For the first simulation we set $s = 0.814$, $r = 0.260$, and start with three concatenated SARMs. The result can be seen in the following figure:

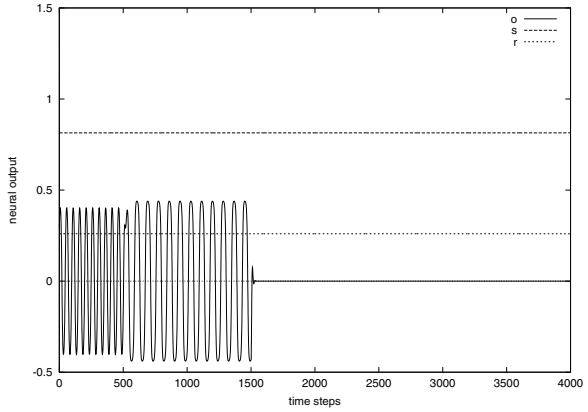

Figure 2: Neural Output When Using Fixed Weights

As expected, the three ring modules oscillate with amplitude $A_0 = 0.4$ and frequency ratio $f_0/f_s = 1/50$, which corresponds to an output frequency of 1.0Hz, assuming 50 time steps per second. After 10 seconds (at time step 500) two additional SARMs are plugged in. In consequence, the amplitude slightly increases and the frequency considerably lowers down from 1.0Hz to 0.6Hz. After further 20 seconds (at time step 1500) four of the five SARMs are removed and almost immediately the oscillation completely breaks down. 20 seconds later again two SARMs are added (at time step 2500), but the system is not able to recover from zero to oscillation. Theoretically oscillation should restart, but due to the finite accuracy (even) of double precision numbers this is practically never the case.

We now repeat the whole sequence of plugging and unplugging, but with learning rules and diffusion processes switched on. In order to have comparable conditions, we use appropriate parameters for target amplitude, target frequency, and network update frequency. As pointed out earlier, it is crucial to quickly reach a stable amplitude, since the frequency controlling adjustment of the ring-connections depends on the amplitude, whereas the amplitude controlling adjustment of the self-connections is frequency independent. Table 2 shows which parameter settings have been proven to be robust. Using these settings we get the following system behavior:

Parameter	Symbol	Value
Target Amplitude	A_0	0.4
Target Frequency	f_0	1Hz
Network Update Frequency	f_s	50Hz
Learning Rate Self-Connections	ε_s	1.0
Learning Rate Ring-Connections	ε_r	5.0
Diffusion Self-Connections	δ_s	0.05
Diffusion Ring-Connections	δ_r	0.1

Table 2: Parameter Settings

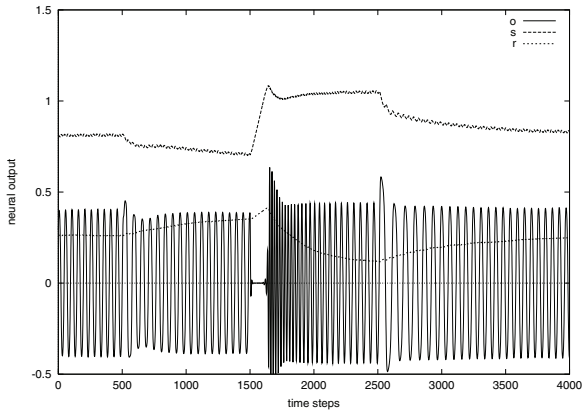

Figure 3: Neural Output of Self-Adjusting Ring Modules

Clearly the SARMs now find back to their equilibrium, after being disturbed by hot reassembly. As can be seen, first the amplitude is back to 0.4 after a few periods, whereas the frequency settles down afterwards (see Figure 3, time steps 500 to 700, 1600 to 2000, and 2500 to 3000). Three findings are to be mentioned in comparison to the simulation with fixed weights:

- Depending on the number of active SARMs, all weights target their optimal value as listed in Table 1. This implicates, that each SARM in principle is able to derive the exact number of attached SARMs – without dedicated communication and regardless of their distance. The starting weights $s = 0.814$ and $r = 0.260$ are reached again, once the starting number of SARMs is restored. Loosely spoken, each part of the body statically feels the rest of the body and is dynamically aware of morphological changes to the whole body.
- The oscillation never breaks down (see time steps 1500 to 1700). More loosely spoken and in analogy to physi-

ology it can be stated, that homeostasis prevents the (artificial) organism from death.

- Close inspection of the slow varying self-connections reveals, that the target frequency shines through slightly (can be best seen shortly after time step 2500). This indicates an optimal setting of the self-connection's learning rate. A higher rate would lead to destabilization, whereas lower rates would prolongate reaction time.

Despite these more analytical and philosophical reflections, the Self-Adjusting Ring Modules can practically be built in hardware and used to make artificial creatures move. An actuated modular body construction system as described in [Raffle, 2004] forms an optimal application. It then is a good idea to have the modules equipped with a switch which introduces a negation before one ring-connection, as necessary for an even number of interconnected ring modules. Now, this switch can also be used to turn off the oscillation of the whole network and settle down into a configuration of constant neural output signals. Different steady output configurations are possible, just by switching on the negation at different modules.

References

[Cybenko, 1989] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signal and Systems*, 2:303–314, 1989.

[Der and Pantzer, 1999] R. Der and T. Pantzer. Emergent robot behavior from the principle of homeokinesis. *Proceedings of the 1st International Khepera Workshop*, 1999.

[Golubitsky *et al.*, 1998] M. Golubitsky, I. Stewart, P.-L. Buono, and J. J. Collins. A modular network for legged locomotion. *Physica D*, 105:56–72, 1998.

[Hikawa, 2003] H. Hikawa. A digital hardware pulse-mode neuron with piecewise linear activation function. *IEEE Transactions on Neural Networks*, 14(5):1028–1037, 2003.

[Hornik *et al.*, 1989] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[Pasemann *et al.*, 2003] F. Pasemann, M. Hild, and K. Zahedi. SO(2)-networks as neural oscillators. *Proc. of Int. Work-Conf. on Artificial and Natural Neural Networks (IWANN)*, 2003.

[Pasemann, 1995] F. Pasemann. Characterization of periodic attractors in neural ring networks. *Neural Networks*, 8:421–429, 1995.

[Raffle, 2004] H. S. Raffle. Topobo: A 3-D constructive assembly system with kinetic memory. Master's thesis, Program in Media Arts and Sciences, MIT, 2004.

[Thompson and Steward, 2002] J. M. T. Thompson and H. B. Steward. *Nonlinear Dynamics and Chaos*. John Wiley & Sons, Ltd, 2002.

[Urzelai and Floreano, 2001] J. Urzelai and D. Floreano. Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evolutionary Computation*, 9(4):495–524, 2001.