Supplementary Materials for:

Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning

Gregor M. Hoerzer[1], Robert Legenstein[1], Wolfgang Maass[1, 2]

[1] Institute for Theoretical Computer Science
Graz University of Technology
A-8010 Graz, Austria
Tel.: +43 316 873-5822
Fax: +43 316 873-5805
[gregor,legi,maass]@igi.tugraz.at
[2] Corresponding author: maass@igi.tugraz.at

Contents:

1. Supplementary Methods

Recurrent neural network model and readout plasticity

For the recurrent network, we used a set of *N=1000* leaky integrator neurons that were sparsely connected in a recurrent fashion.

Connection probabilities and weight distributions: Neurons in the recurrent network were connected with a probability *p=0.1*. In other words, the strength W_{ij}^{rec} of each recurrent connection was set to zero with probability *(1-p)*. The weights of the network connectivity matrix W^{rec} were drawn from a Gaussian distribution with zero mean and a variance of *1/(pN)*. The input weights W^{in} and feedback weights W^{fb} were drawn from a uniform distribution in the interval *[-1, 1]*. The output weights $\mathbf{w}_i$, which we adapted during learning, were initialized to zero in our simulations. In principle, they can also be set to nonzero values. However, these values need to be relatively small in order to avoid that the readout output is too sensitive to noise on the network state because large values need to cancel each other out in order to obtain a small readout output. Note also that while the readouts with feedback in our simulations receive input and also project their output to all neurons within the network, each readout neuron could in principle also be substituted by multiple neurons with sparse connectivity.

In our simulations, we used the following parameter settings: network size *N=1000* units, internal connectivity *p=0.1*, time constant $\tau = 50\text{ms}$ for the first and $\tau = 10\text{ms}$ for the other simulation tasks, chaoticity level $\lambda = 1.5$ for the first simulation task (without input) and $\lambda = 1.2$ for the other (partly input driven) simulation tasks, simulation time step $\Delta t = 1\text{ms}$.

Noise distributions: The noise $\xi_i^{\text{state}}(t)$ in the firing rate $r_i(t) = \tanh(x_i(t)) + \xi_i^{\text{state}}(t)$ of neuron *i* was drawn from a uniform distribution in the interval $[-\theta^{\text{state}}, \theta^{\text{state}}]$ at each time step, with θ^{state} =0.05. Since the sigmoidal $\tanh(\cdot)$ function nonlinearly maps the neuron's state $x_i(t)$ onto the interval *[-1, 1]*, the neuronal output $r_i(t)$ assumed values in the interval $[-1 - \theta^{\text{state}}, 1 + \theta^{\text{state}}]$. The exploration noise $\xi(t)$ was drawn independently for each readout and time step from a uniform distribution in the interval *[-0.5, 0.5]*.

In a subset of simulations, we used temporally correlated exploration noise. In these simulations, the actual noise at time *t* was given by

$$\xi(t) = \left(1 - \frac{\Delta t}{\tau_{\text{noise}}}\right)\xi(t - \Delta t) + \left(\frac{\Delta t}{\tau_{\text{noise}}}\right)X(t), \quad \text{(S1)}$$

where Δt is the simulation time step and τ_{noise} is the time constant of the temporal correlation. It was chosen between 1ms (corresponding to no noise correlation since Δt =1ms) and 5ms. X denotes a random variable that is drawn independently in every time step from a uniform distribution over the interval *[-0.5, 0.5]*. The initial value was also drawn from this distribution.

Decaying learning rate: We used a slowly linearly decaying learning rate of

$$\eta(t) = \frac{\eta_{\text{init}}}{1 + \frac{t}{T}} \quad \text{(S2)}$$

with initial learning rate η_{init} and a decay constant T*=20s*. This linear decay ensures that adaptations take place throughout the whole learning interval and satisfies the convergence conditions for stochastic approximation (Robbins and Monro, 1951). The initial learning rate was $\eta_{\text{init}} = 0.0005$ for the reward modulated Hebbian learning rule, and the initial learning rate $\eta_{\text{init}} = 0.0001$ for the LMS-based FORCE rule (for information on the choice of the learning rate see Supplementary Results below). Additional simulations were performed with a constant learning rate (see *Supplementary Results*).

Filter operations on readout output and performance: For the reward-modulated Hebbian learning rule, the running average of the noisy readout output was given by

$$\bar{z}_i(t) = \left(1 - \frac{\Delta t}{\tau_{\text{avg}}}\right)\bar{z}_i(t - \Delta t) + \left(\frac{\Delta t}{\tau_{\text{avg}}}\right)z_i(t), \quad \text{(S3)}$$

and the running average of the overall performance was given by

$$\bar{P}(t) = \left(1 - \frac{\Delta t}{\tau_{\text{avg}}}\right)\bar{P}(t - \Delta t) + \left(\frac{\Delta t}{\tau_{\text{avg}}}\right)P(t), \quad \text{(S4)}$$

where Δt is the simulation time step. We used τ_{avg}*=5ms* in all simulations except for those simulations where τ_{avg} was varied.

Definitions of target functions for the simulations

The periodic function which had to be produced in the first simulation task was

$$f(t) = 1.3/1.5\sin(2\pi 1t) + 1.3/3\sin(2\pi 2t) + 1.3/9\sin(2\pi 3t) + 1.3/3\sin(2\pi 4t). \quad \text{(S5)}$$

The signal of the pulse input streams of the second and third simulation task were smoothed 100ms pulses which were generated with an average rate of 0.5Hz. More precisely, two variables $\hat{u}_{\text{on},i}(t)$ and $\hat{u}_{\text{off},i}(t)$ that indicated the start of a pulse in the on and off stream respectively, were independently set to 1 with a probability of 0.0005, and to 0 otherwise in each time step Δt. The input streams $u_{\text{on},i}(t)$ and $u_{\text{off},i}(t)$ were then smoothed pulses with an amplitude of 0.4 and a duration of 100ms (before smoothing). The smoothing filter time constant was 50ms. More formally, $u_{\text{on},i}$ was given by $u_{\text{on},i} = \frac{1}{\sigma_u}(\Theta_0 \circ (\hat{u}_{on,i} * h)) * g$, where $*$ denotes the convolution operation (we used a discrete convolution where the functions g and h were discretized with time-steps Δt), $\circ$ denotes function composition, and $\Theta_x(s)$ is the Heaviside function that is 0 for $s<0$, x for $s=0$, and 1 otherwise. $h(s) = \Theta_1(s) - \Theta_1(s - 100ms)$ is a 100ms pulse, and $g(s) = \exp(-s/\tau_L)\Theta_1(s)$ is an exponential smoothing function with time constant $\tau_L = 50ms$. The constant σ_u scales the pulses to an amplitude of 0.4. $u_{\text{off},i}$ was produced in the same way from $\hat{u}_{\text{off},i}$.

The target values for the memory units were -0.5 for the ``off" state and 0.5 for the ``on" state:

$$\hat{f}_i(t) = \begin{cases} 0.5 & if \quad \hat{u}_{\text{on},i}(t) = 1, \\ -0.5 & if \quad \hat{u}_{\text{off},i}(t) = 1, \\ \hat{f}_i(t - \Delta t) & otherwise\,. \end{cases} \tag{S6}$$

The actual target function $f_i = \frac{1}{\sigma_f}\hat{f}_i * g$ was then a smoothed version of $\hat{f}_i(t)$ with the exponential filter g as given above and a time constant τ_{L} of 50ms. Here, σ_f denotes a constant scaling factor that rescaled the function such that the values of $f_i(t)$ ranged between -0.5 and 0.5 again after the application of the smoothing filter.

The values of the two independent temporally correlated analog inputs $u_3(t)$ and $u_4(t)$ in the third simulation task were each drawn from a uniform distribution in each time step, filtered with a time constant of 0.5s and then scaled to have a standard deviation of 0.25. Bias values of 0.3 and 0.15 were added to these inputs, respectively.

The target function $f_2(t)$ was then defined as

$$f_2(t) = \begin{cases} u_3(t) & if \quad \hat{f}_1(t) > 0, \\ u_4(t) & otherwise\,. \end{cases} \tag{S7}$$

Comparison to FORCE Learning

Because we compare the performance of the reward modulated learning rule to the performance of the supervised FORCE learning rules proposed by (Sussillo and Abbott, 2009), we state them here in short. In contrast to the learning rule used here, the two variants of the FORCE learning mechanism use the exact error $e_i(t) = \hat{z}_i(t) - f_i(t)$ of the i-th readout to update its weights $\mathbf{w}_i$. In the local least mean squares (LMS) based FORCE rule, readout weights are adapted according to

$$\Delta \mathbf{w}_i(t) = -\eta(t) e_i(t) \mathbf{r}(t). \tag{S8}$$

The more powerful recursive least squares (RLS) based FORCE rule is defined as

$$\Delta \mathbf{w}_i(t) = -e_i(t) C(t) \mathbf{r}(t), \tag{S9}$$

the matrix *C(t)* being a running estimate of the inverse of the correlation matrix of the network output **r***(t)* plus a regularization term (cf. Haykin, 2001; Sussillo and Abbott, 2009 for details). Therefore, the RLS-based FORCE rule uses information about the activation of all synapses to the readout neuron to modify the learning rate of an individual synapse. This procedure makes the learning rule more powerful than the simple LMS-based rule, but at the expense of locality and simplicity of the rule, which involves some rather complicated computations to generate the matrix *C(t)*.

For consistency with (Sussillo and Abbott, 2009) we did not apply exploration noise during learning of the FORCE-trained systems. However, control simulations with noise showed that this noise does not significantly change the system behavior (data not shown).

Performance evaluation

To evaluate the performance of the trained system for different parameter settings in the periodic trajectory production task, we used the following procedure. Since there are no input signals in this simulation task (cf. Figure 1A), the readout has no reference during the testing period, and small errors in the frequencies of the trained signal components therefore lead to a varying shift between the target and the readout output over time, i.e. the actual oscillation cycle length of the output of the readout is slightly longer or shorter than the one of the target signal (cf. Figures 1D and 1H). To see whether the shape of the target signal is nevertheless accurately reproduced by the readout's

output, we cut the readout's output during the testing interval into successive time slices of one second, which corresponds to the oscillation cycle length of the target function *f(t)*. Then, we calculated the minimum mean squared error (MSE) between each time slice and circularly shifted versions of a one-cycle slice of the target signal, instead of just calculating the MSE between the target function *f(t)* and the readout's output $\hat{z}(t)$ directly.

In the comparative simulations employing the FORCE learning procedures, we conducted simulations with and without adding exploration noise during learning. The traces shown in figure 1E correspond to results without added exploration noise during learning. In any case, the error e_i of the i-th readout that was used for weight adaptation was based on the readout output without exploration noise in these simulations.

To evaluate the performance of the memory units in the ``persistent memory" and the ``switchable routing" task, we calculated the percentage of time steps during the testing interval in which the absolute difference between the target function and the output of the readout neuron exceeded the threshold 0.5 (half the difference between the two target values 0.5 and -0.5 for the two different states).

2. Supplementary Results

The supplementary results section discusses only those results that have not been discussed in the main article. Since we arranged the figures in the order in which they are referenced in the main text and also grouped panels that are referenced in the main text together with non-referenced panels that show similar types of simulation results into single figures, only a subset of the figure panels are discussed here.

Additional tests for noise robustness of the system

To test the robustness of the system, we perturbed the network in different ways during learning and testing. In order to observe the response of the system to different amounts of noise on the network state during learning, we performed simulations with state noise levels θ^{state} of 0.0, 0.05 (as in the original simulations), 0.1, 0.2, 0.3, 0.4, 0.5, 0.75 and 1.0 applied during the learning period. During testing, we always used the original state noise level of 0.05 in order to keep the results comparable. The learning time was 400s in all simulations. The mean squared error (MSE) was calculated across the whole testing period of 500s as described in the performance evaluation section above. Based on the MSE, we calculated median values across 50 trials per state noise level. The results are shown in Supplementary Figure S2E. Our simulations revealed that the system is very robust against noise perturbations of the network state during learning, leading to significant decreases in system performance only for surprisingly high amounts of state noise.

To further test the robustness of the system to unexpected perturbations during the testing interval, we applied the following perturbations and observed the system response. First, the system was perturbed for one second by noise in the feedback signal of the same amplitude as the exploration noise applied during learning. Second, it was perturbed with state noise on the network output $\mathbf{r}(t)$ that was ten times stronger than the state noise usually used in the simulations. Figures S2F and S2G show that the trained system is robust against perturbations of the feedback signal (panel F) that affect the readout only via the network, and perturbations of the network output (panel G), which directly affect the input to the readout neuron itself. While perturbations of the feedback signal have no visible effect on the output of the readout since they are filtered by the slow dynamics of the network, the strong perturbations of the network output do have a visible effect on the readout output. However, while this type of noise lets the readout fluctuate around the desired trajectory during the time interval in which the system is perturbed, the desired trajectory is

restored shortly after the noise is removed again. Interestingly, the network is able to restore the desired trajectory even after extremely severe perturbations of the readout output. In a third perturbation simulation, we clamped the readout output such that it remained constant for 0.5s. After unclamping of the readout output, the system is able to recover the desired trajectory within a few oscillation cycles (see panel H of Figure S2).

Choice of the learning rate for the EH rule and LMS FORCE

In order to find an appropriate learning rate, we simulated the system with different learning rates η between 0.001 and 0.00001 for both the LMS FORCE (Supplementary Figure S4A, left panel) and the reward modulated Hebbian learning method (right panel). The results indicate that the system performed best using a learning rate of η=0.0001 for the LMS FORCE rule and η=0.0005 for the reward modulated Hebbian learning rule. Performance increased for decreasing learning rates up to the learning rate used for our comparison between our learning rule and the FORCE rules. For even lower learning rates, the system failed to adapt quickly enough in order to follow the target trajectory appropriately after the beginning of the learning process, leading to significantly decreased performance for too small learning rates. We also performed simulations with and without 'exploration noise' applied during learning for the LMS FORCE rule (which is not needed for this kind of learning). The resulting performances were similar in both cases.

Simulations with constant learning rate and analog modulatory signal

We consistently used decaying learning rates and a binary modulatory signal for the results reported in the main text. Here we report simulations with constant learning rates and an analog modulatory signal for the weight updates. The comparison shows that the system performs equally well in all three conditions. In our simulations with constant learning rate, we did not decay the initial learning rate over time. For the analog performance-signal case, we used $P - \bar{P}$ directly instead of using the binary signal *M(t)* in the weight update rule (7) together with a decaying learning rate. We performed simulations with η=0.0005 and η=0.001 for both conditions. Supplementary Figure S4B shows the simulation results, indicating that the performance does not vary significantly across the three conditions (decaying learning rate and binary modulatory signal, constant learning rate and binary modulatory signal, decaying learning rate and analog modulatory signal).

Simulations with different target patterns

In order to investigate a possible relation of the necessary learning time and the complexity of the target pattern, we performed simulations with target patterns with different numbers of superimposed frequency components (again, 50 simulation trials per learning time and pattern type). We used the same amplitudes for the frequency components as in the original target pattern, but left out some components for the simulations with lower numbers of frequency components. Supplementary Figure S4C shows the results, indicating that a simple sinusoid can be learned very quickly. For the target patterns with two to four superimposed sinusoidal components, the learning time was similar to each other. For five superimposed sinusoidal components (1-5Hz, in 1Hz steps), the system failed to reach the same level of performance as for the less complex target patterns.

Furthermore, we compared the results of the reward modulated Hebbian learning rule with the LMS FORCE rule for different target patterns. These patterns had the same number of superimposed components as the original target pattern (1-4Hz, in 1Hz steps), but with different amplitudes of the frequency components. This leads to target patterns with different shapes (see Supplementary Figure S4D). Our results indicate that performance can in general be different across different target patterns, but is nevertheless similar for both learning rules.

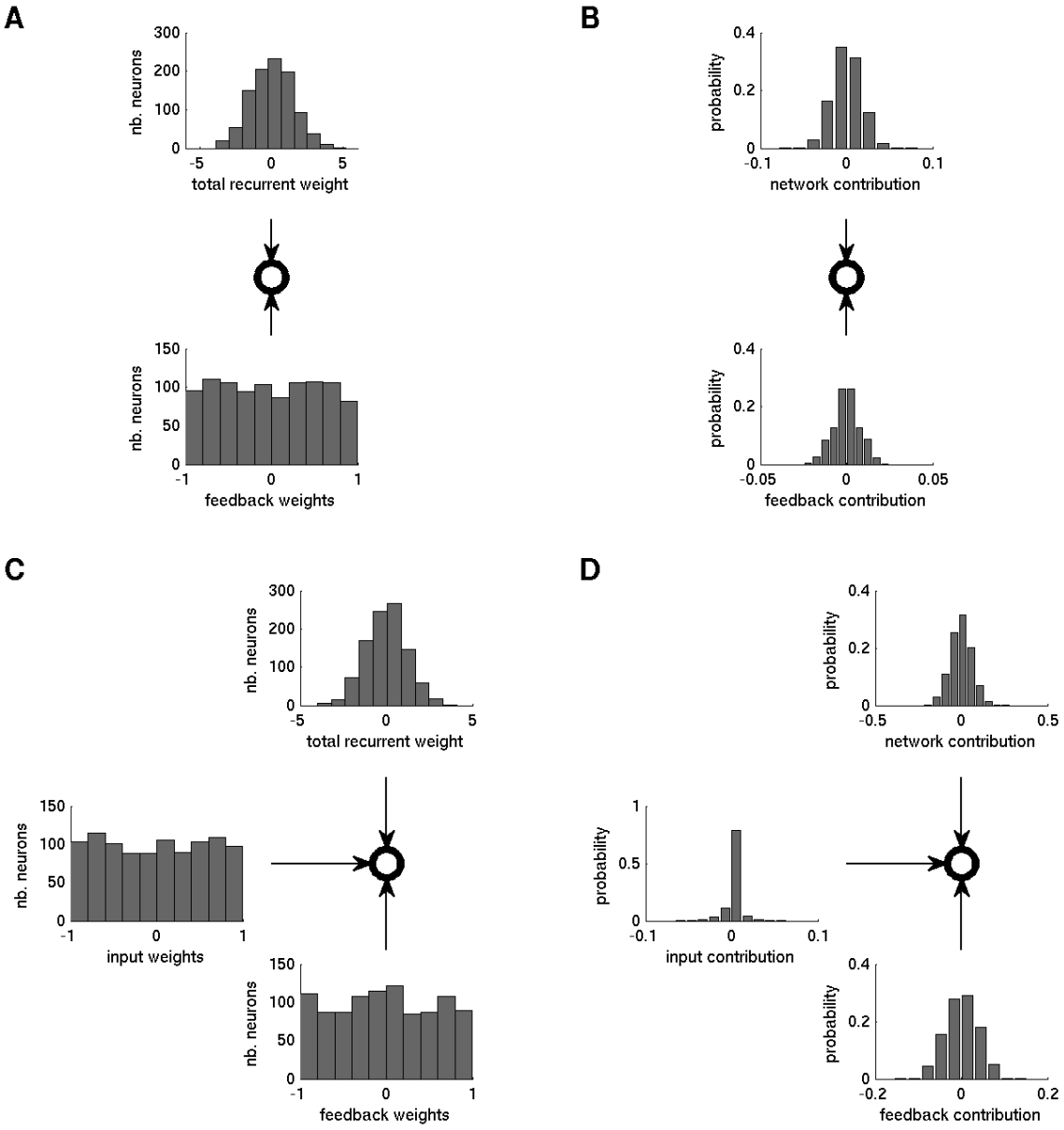
References

Haykin S (2001) Adaptive Filter Theory (4th Edition): Prentice Hall.

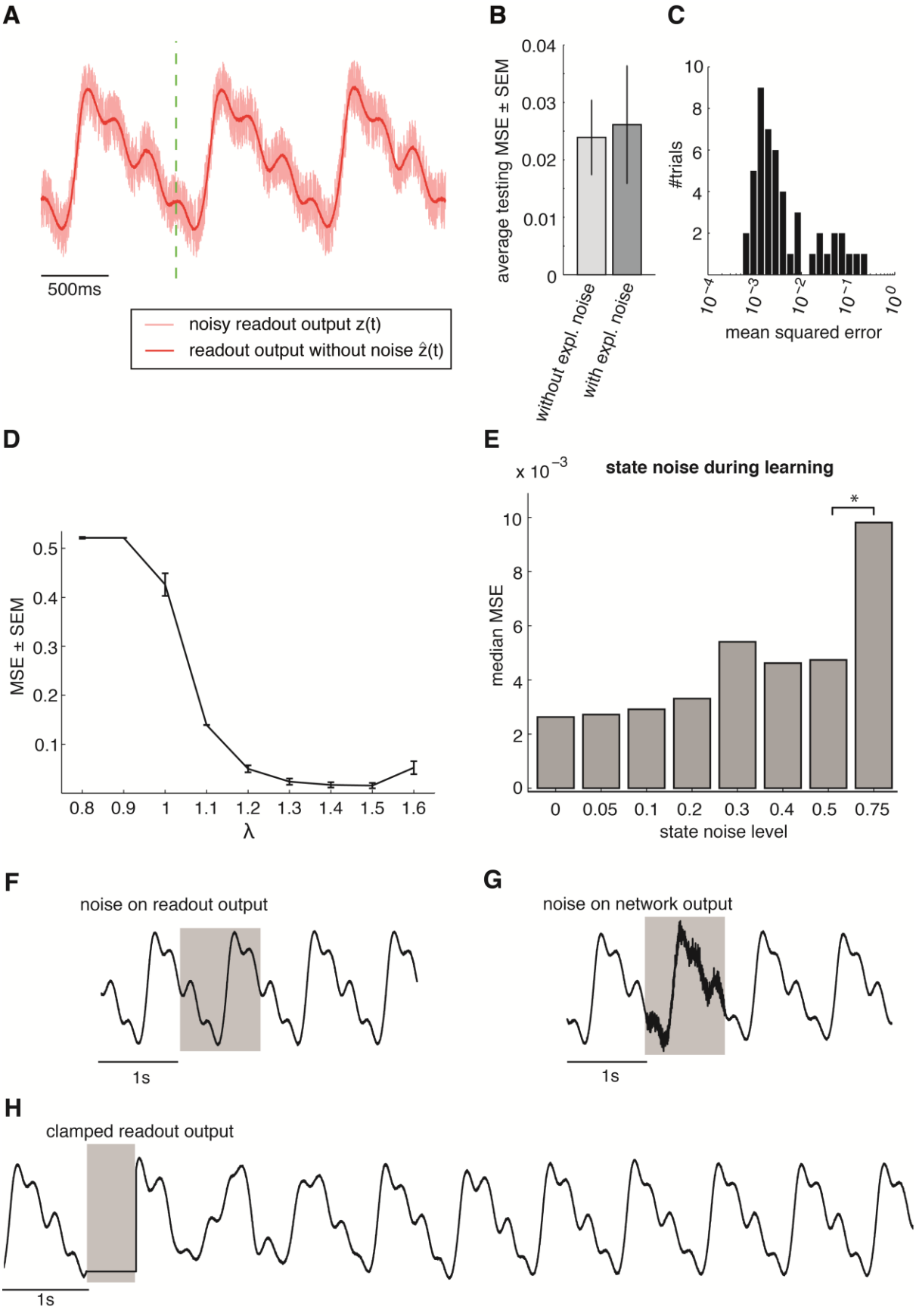
Robbins H, Monro S (1951) A Stochastic Approximation Method. The Annals of Mathematical Statistics 22:400-407.

Sussillo D, Abbott LF (2009) Generating Coherent Patterns of Activity from Chaotic Neural Networks. Neuron 63:544-557.

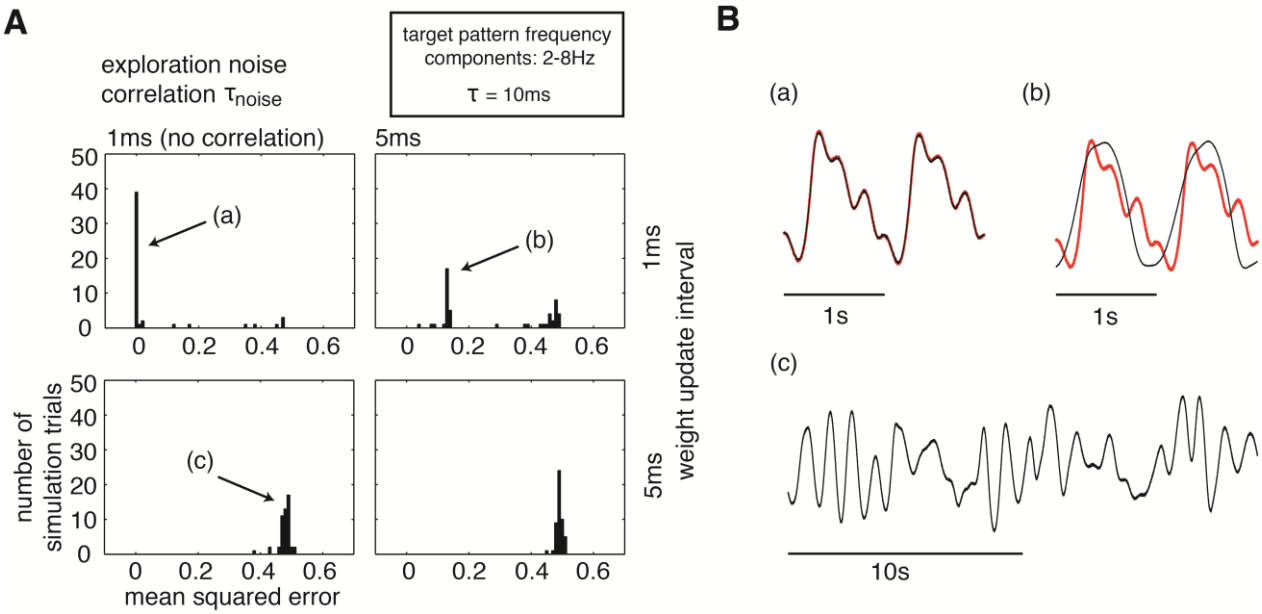
3. Supplementary Figures



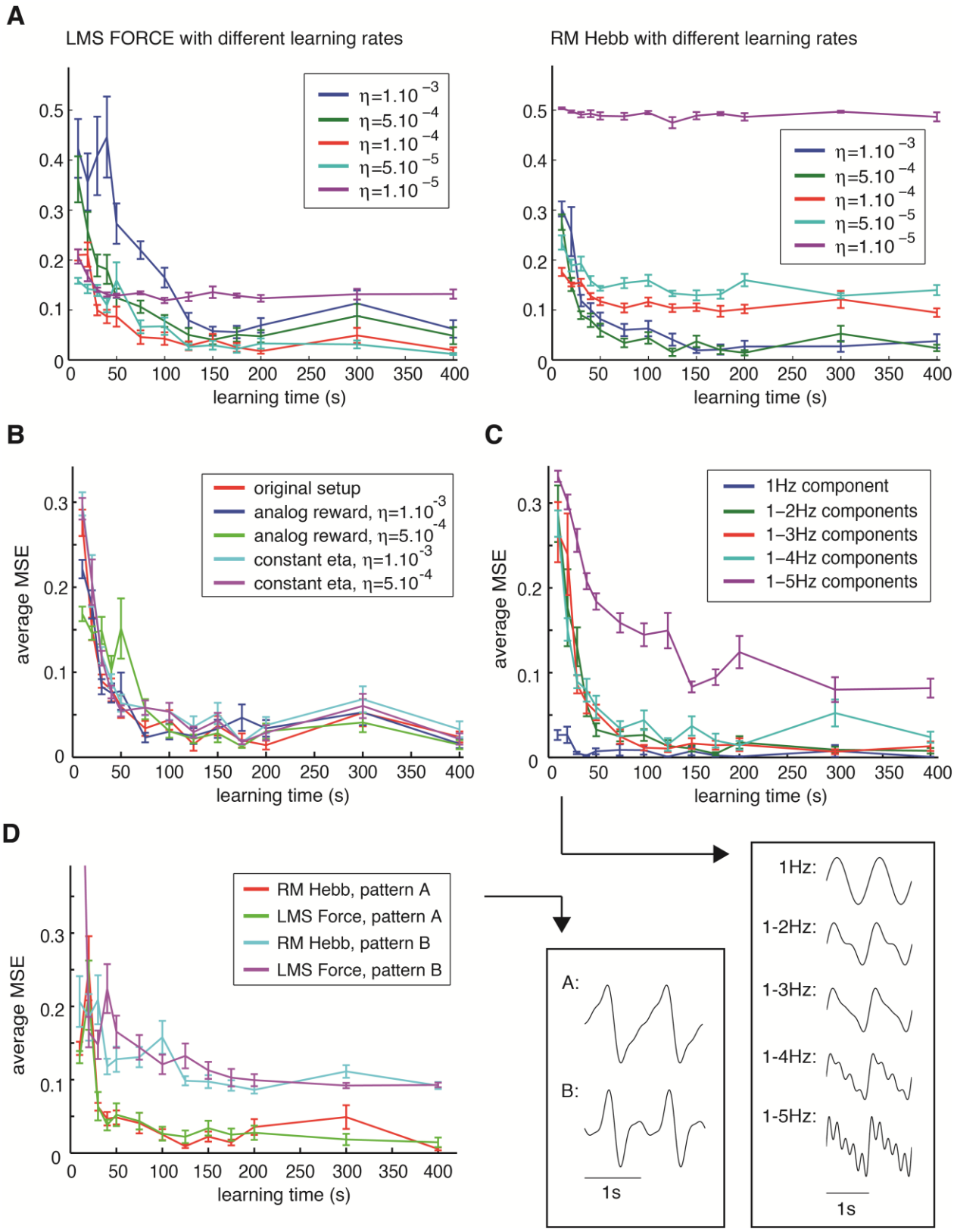
Supplementary Figure S1: Contributions of recurrent connections, input connections, and feedback connections to the membrane potential of network neurons. Shown are contributions for two tasks considered in this article, a peridoc pattern generation task (A, B) and a working memory task (C,D). **A:** Histogram of the number of neurons with a given feedback weight (bottom) and a given total weight from recurrent connections (top). The total weight is the sum over all incoming weights from recurrent network neurons (this includes the scaling factor λ). **B:** Histogram of the contributions of feedback connections (bottom) and recurrent connections (top) to the membrane potential $x_i(t)$ of network neurons. Probability of occurrence was computed over all time-steps and all network neurons in a 5 sec. test period after 500 sec. learning. **C, D:** Same as A and B respectively, but for a working memory task. In this task, input is provided to network neurons. The middle histograms show the number of neurons with a given weight from one of the input connections (C) and the contributions of input connections to the neuron's membrane potential $x_i(t)$ (D). The sparse activity of pulse inputs in the working memory task is reflected in a sparse distribution in this histogram.


A
500ms
noisy readout output z(t)
readout output without noise ẑ(t)
B
average testing MSE ± SEM
0.04
0.03
0.02
0.01
0
without expl. noise
with expl. noise
C
#trials
10
8
6
4
2
10^-4
10^-3
10^-2
10^-1
10^0
mean squared error
D
MSE ± SEM
0.5
0.4
0.3
0.2
0.1
0.8 0.9 1 1.1 1.2 1.3 1.4 1.5 1.6
λ
E
x 10^-3
state noise during learning
*
median MSE
10
8
6
4
2
0
0 0.05 0.1 0.2 0.3 0.4 0.5 0.75
state noise level
F
noise on readout output
1s
G
noise on network output
1s
H
clamped readout output
1s

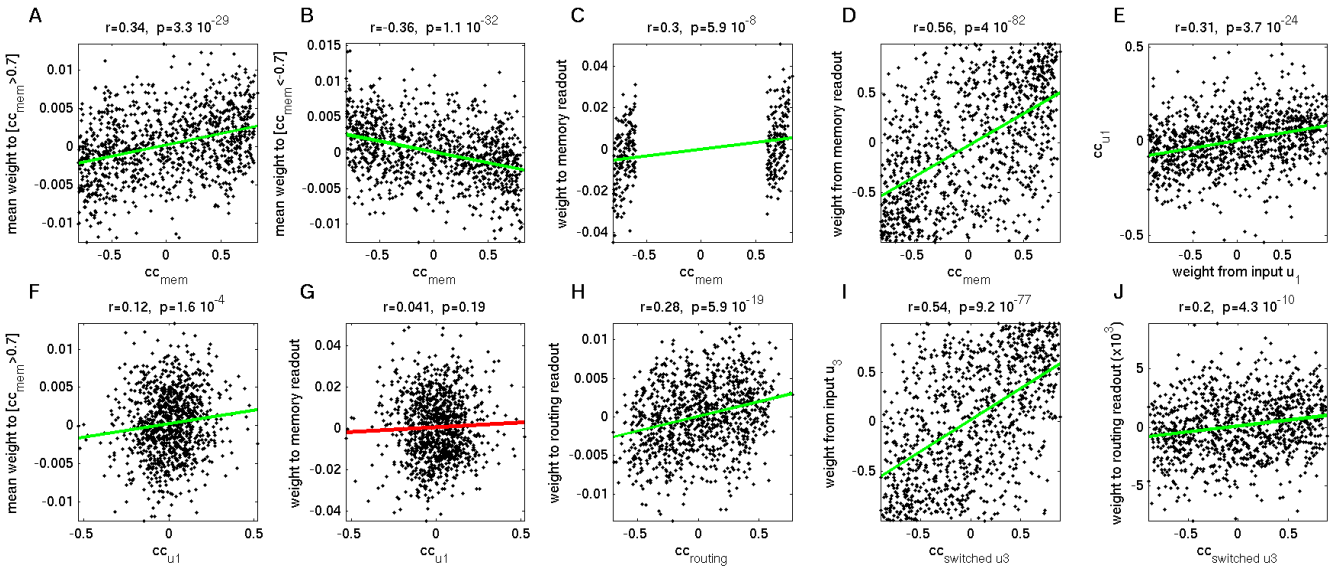
Supplementary Figure S2: Relation between chaoticity of the network, network performance, and robustness of the system against perturbations. A: System performance with exploration noise also applied during testing. Example trace of the readout output at the beginning of the testing period after 400s of learning. Although the feedback from the readout to the network is noisy also during testing, the readout keeps producing the desired trajectory. **B:** Comparison of the mean squared error (MSE) during a testing period of 500s without (left) and with (right) added exploration noise applied during testing (average over 50 independent simulations, 400s of learning). The system performance is similar in both cases. **C:** Distribution of the MSE across simulation trials within the last second at the end of a 500s testing interval (log scale) for the reward modulated Hebbian learning rule after 400s of learning (cf. Figure 1G, red trace). The network keeps producing the target function properly until the end of testing in most of the trials. **D:** Performance of the network after 400s of learning for different values of λ. The network needs a certain level of chaoticity, leading to rich enough dynamics, in order to generate the target function without further input. The system performs best for λ values between 1.3 and 1.5. **E:** Median MSE for different state noise levels applied during learning. The value 1.0 was omitted here because it was much larger than the other noise levels (value: 0.1393). We compared neighboring noise levels for significant differences in performance distribution across trials. Only the distributions of the two largest levels (0.75 and 1.0) were significantly larger than performance of the the next lower level (as indicated by the bar with the star at for the level of 0.75). (nonparametric Wilcoxon rank sum test, significance level: 5%). **F, G:** Noise robustness of the trained trajectory. Uniformly distributed noise (of the same amplitude as the exploration noise used during learning) was added to the feedback from the readout $\hat{z}(t)$ (panel F, noise added in the shaded 1s interval) as well as to the network output $\mathbf{r}(t)$, which represents the input to the readout unit (panel G, noise added in the shaded 1s interval). Noise on the readout output does not produce any visible deviations of the readout output. Noise on the network output directly affects the input to the readout and produces deviations that fluctuate around the desired trajectory. The desired trajectory is quickly restored after the noise is removed again. **H:** The readout output was clamped to a constant value for a time interval of 0.5s. After unclamping the output of the readout, the system is able to restore the desired trajectory within a few oscillation cycles.



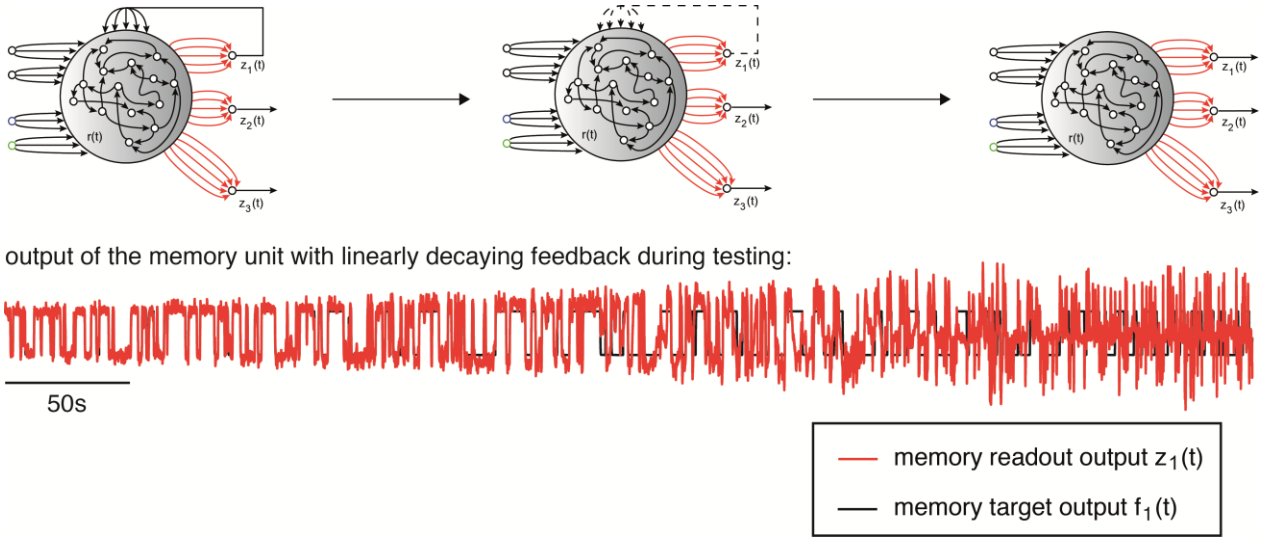
Supplementary Figure S3: Simulations using different time constants for the network dynamics. A: Given that weight updates are fast (1ms) and the exploration noise is not temporally correlated, the system is also able to perform well for patterns with frequency components of up to 8Hz if the time constant of the network dynamics is reduced to τ=10ms (cf. Figure 2A, upper right and middle right panels). For slower weight updates and for temporally correlated exploration noise, the system fails to produce the target function. **B:** Representative example traces of the readout activity (black) that show the system behavior in simulation trials with varying performance during the testing period and target output (red). The subpanel indices (a) to (c) are being indicated by the indices with arrows in panel A.


A
LMS FORCE with different learning rates
RM Hebb with different learning rates
η=1.10 −3
η=5.10 −4
η=1.10 −4
η=5.10 −5
η=1.10 −5
learning time (s)
B
original setup
analog reward, η=1.10-3
analog reward, η=5.10-4
constant eta, η=1.10-3
constant eta, η=5.10-4
average MSE
C
1Hz component
1–2Hz components
1–3Hz components
1–4Hz components
1–5Hz components
D
RM Hebb, pattern A
LMS Force, pattern A
RM Hebb, pattern B
LMS Force, pattern B
A:
B:
1s
1Hz:
1-2Hz:
1-3Hz:
1-4Hz:
1-5Hz:

Supplementary Figure S4: Additional simulations. A: Comparison of different learning rates for the LMS FORCE (left) and Reward-modulated Hebbian (right) learning rules. We chose a learning rate of η=0.0001 (left, red trace) for the LMS FORCE rule and η=0.0005 (right, green trace) for the reward modulated Hebbian rule for our comparison between the learning rules, showing the respective best performance across different values of η. In general, the reward modulated Hebbian rule needed higher values of η in order to perform properly. **B:** Simulations with analog reward and constant learning rate η, for two different learning rates each. In all cases, the system was able to perform the task properly if the system was trained for a sufficiently long time. **C:** Simulations with target patterns with different number of superimposed frequency components. While the system is able to learn a simple 1Hz oscillation within a few seconds of learning, it needs to be trained for a longer, but approximately similar time for two to four superimposed frequency components. For the simulations with five superimposed frequency components, the system in general showed decreased performance. **D:** Simulations with other target patterns with both LMS Force learning and our reward modulated Hebbian learning rule. For both patterns A and B, while having different performances for the different patterns, both learning rules performed similar for learning times larger than 75s in all but one cases (nonparametric Wilcoxon rank sum tests, significance level: 5%, exception: pattern B, 300s learning time, p=0.0172).



Supplementary Figure S5: Correlations between network neuron behavior and weight parameters in the switchable routing task. A: The correlation coefficient a neurons activity with the target function of the memory readout cc_{Mem} is correlated with the mean weight of this neuron to neurons with high cc_{Mem}. **B:** The correlation coefficient a neurons activity with the target function of the memory readout cc_{Mem} is anti-correlated with the mean weight of this neuron to neurons with large negative cc_{Mem}. **C:** The weight of a neuron to the memory readout is significantly correlated with its cc_{Mem} for neurons with $|cc_{Mem}| > 0.6$. No such correlation is seen if all neurons are taken into account. **D:** cc_{Mem} is strongly correlated with the feedback weight of the memory readout to this neuron. **E:** The weight that a neuron receives from pulse input u_1 is correlated with the correlation coefficient cc_{u1} between this neurons activity and u_1. **F:** cc_{u1} is slightly correlated with the mean weight of this neuron to neurons with large cc_{Mem}. **G:** The correlation between the weight of a neuron to the memory readout and its cc_{u1} is not significant. **H:** The correlation coefficient $cc_{routing}$ of a neurons activity with the target function for the routing readout is correlated with this neurons weight to the routing readout. **I:** $cc_{switched\ u3}$ (the correlation coefficient of a neurons activity with the target function for the routing readout at times when u_3 should be routed to the readout) is strongly correlated with the weight from u_3 to this neuron. **J:** $cc_{switched\ u3}$ is correlated with this neurons weight to the routing readout.



Supplementary Figure S6: Output of the memory unit in the switchable routing task with feedback weights linearly decaying to zero after learning. While the system is able to keep the memory states stable for a surprisingly long time despite the decaying feedback, the amplitude of the output is amplified rather than reduced, indicating a regulatory role of the feedback from the memory readout.