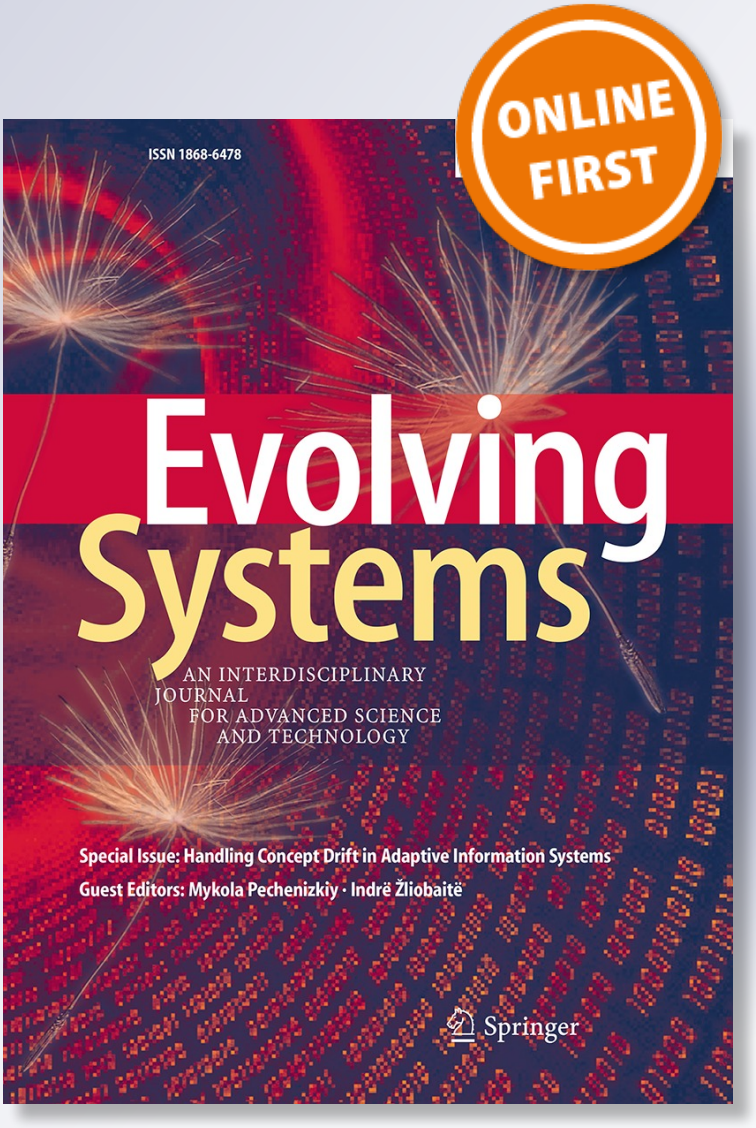
# *Information dynamics based self-adaptive reservoir for delay temporal memory tasks*

**Sakyasingha Dasgupta, Florentin Wörgötter & Poramate Manoonpong**

**Evolving Systems**
An Interdisciplinary Journal for Advanced Science and Technology

ISSN 1868-6478

Evolving Systems
DOI 10.1007/s12530-013-9080-y




Springer

Evolving Systems
DOI 10.1007/s12530-013-9080-y

ORIGINAL PAPER

# Information dynamics based self-adaptive reservoir for delay temporal memory tasks

**Sakyasingha Dasgupta · Florentin Wörgötter · Poramate Manoonpong**

Received: 27 December 2012 / Accepted: 23 April 2013


**Abstract** Recurrent neural networks of the reservoir computing (RC) type have been found useful in various time-series processing tasks with inherent non-linearity and requirements of variable temporal memory. Specifically for delayed response tasks involving the transient memorization of information (temporal memory), self-adaptation in RC is crucial for generalization to varying delays. In this work using information theory, we combine a generalized intrinsic plasticity rule with a local information dynamics based schema of reservoir neuron leak adaptation. This allows the RC network to be optimized in a self-adaptive manner with minimal parameter tuning. Local active information storage, measured as the degree of influence of previous activity on the next time step activity of a neuron, is used to modify its leak-rate. This results in RC network with non-uniform leak rate which depends on the time scales of the incoming input. Intrinsic plasticity (IP) is aimed at maximizing the mutual information between each neuron's input and output while maintaining a mean level of activity (homeostasis). Experimental results on two standard benchmark tasks confirm the extended performance of this system as compared to the static RC (fixed leak and no IP) and RC with only IP. In addition, using both a simulated wheeled robot and a more complex physical hexapod robot, we demonstrate the ability of the system to achieve long temporal memory for solving a basic T-shaped maze navigation task with varying delay time scale.

**Keywords** Recurrent neural networks · Self-adaptation · Information theory · Intrinsic plasticity · Temporal memory

S. Dasgupta (✉) · F. Wörgötter · P. Manoonpong
Bernstein Center for Computational Neuroscience (BCCN), Department of Computational Neuroscience, University of Göttingen, Friedrich-hund Platz 1, 37077 Göttingen, Germany
e-mail: s.dasgupta@physik3.gwdg.de

F. Wörgötter
e-mail: worgott@physik3.gwdg.de

P. Manoonpong
e-mail: poramate@physik3.gwdg.de

## 1 Introduction

Reservoir computing (RC) is a powerful paradigm for the design, analysis and training of recurrent neural networks (Lukosevicius and Jaeger 2009). The RC framework has been utilized for mathematical modeling of biological neural networks (Maass et al. 2004) as well as applications for non-linear time-series modeling (Shi and Han 2007), robotic applications and understanding the dynamics of memory in large recurrent networks in general (Büsing et al. 2010). Traditionally the reservoir is randomly constructed with only the output connections trained with a regression function. Although both spiking and analog neurons have been explored previously, here we focus on the Echo-state network (ESN) type (Jaeger and Haas 2004) using sigmoid leaky integrator neurons.

Even though the generic RC shows impressive performance for many tasks, the fixed random connections and variations in parameters like spectral radius, leak-rate and number of neurons can lead to significant variations in performance. Approaches based on intrinsic plasticity (IP) (Schrauwen et al. 2008) can help to improve such generic reservoirs. IP uses an information theoretic approach for information maximization at an individual neuron level in a self-organized manner. The IP performance significantly depends on the type of transfer function, degree of sparsity required and the use of different probability distributions.

Published online: 12 May 2013

However the conventional IP method is still outperformed by specific network connectivities like permutation matrices, in terms of the memory capacity performance (Boedecker et al. 2009).

Here we overcome this, by first utilizing a new IP method (Li 2011) based on a Weibull distribution for information maximization. This is then combined with an adaptation rule for the individual neuron leak-rate based on the local information storage measure (Lizier et al. 2011, 2012). Transfer entropy is another measure for such an adaptation rule. However conventionally this is more difficult to compute, and as it also maximizes input to output information transfer, it is difficult to combine with an IP rule. We achieve such a combination in a self-organized way to guide the individual neurons for both, maximizing their information content and their local memory based on the incoming input signal. Subsequently through two standard benchmark tasks and the robot maze navigation tasks, we show that our adapted network has better performance and memory capacity as compared to static and only IP adapted reservoirs. All the tested scenarios involve a high degree of non-linearity and requirement of adaptable temporal memory. Specifically in robotics and engineering control tasks with nonlinear dynamics and variational inputs (in the time domain), our adaptation technique can show significant performance.

This article is organized as follows. Section 2 describes the self-adaptive reservoir framework together with the network dynamics and the two adaptation rules, namely intrinsic plasticity and the local information storage based leak adaptation. Section 3 presents the experimental setup and analysis. Section 4 presents the experimental results and illustrates the performance of our network for standard benchmark tests (NARMA-30[1] and delayed 3-bit parity) and for the maze navigation task for both simulation and real robot scenarios. Section 5 discusses the presented framework in general along with biological relevance of our network in terms of timing mechanisms in the brain and memory guided behaviors. This is followed by the conclusion in Sect. 6.

## 2 Self-adaptive reservoir framework

In this section we present the description of the internal reservoir network dynamics and introduce (i) neuron local memory adaptation based on active information storage measure and (ii) the self-organized adaptation of reservoir neurons inspired by intrinsic plasticity. These are carried out as unsupervised rules as part of the pre-training phase of the reservoir network. Subsequently, we combine both mechanisms for a comprehensive adaptive framework.

### 2.1 Network description

The recurrent neural network (RNN) model based on the reservoir computing framework is depicted in Fig. 1. To a certain extend the model could be considered as an abstract representation of the mammalian neo-cortex. The basic framework can be divided into three layers: input, hidden (or internal) and output layers. The internal layer has a large recurrent neural network that is driven by temporal signals. These driving signals are provided by the input layer. Due to the dynamic reservoir, the network exhibits a wide repertoire of nonlinear activity. This is then combined into desired output signals at the output layer, using a suitable supervised training of the reservoir neuron to output connectivity. The firing activity of the dynamic reservoir at discrete time $t$ is described by the internal state activation vector $\mathbf{x}(t)$. Each neuron is connected to itself or other neurons via weighted synaptic connections. Specifically $\mathbf{W}_{in}$ are the $K \times N$ connections from the $K$ input neurons to the $N$ reservoir neurons, $\mathbf{W}_{out}$ are the $N \times L$ connections from the reservoir neurons to the $L$ output neurons and $\mathbf{W}_{sys}$ represents the $N \times N$ dynamic reservoir recurrent connections.

The recurrent neural activity within the dynamic reservoir varies as a function of it's previous activity and the current driving input signal. As such, the discrete time state dynamics of reservoir neurons is given as:

$$\mathbf{x}(t+1) = (\mathbf{I} - \mathbf{\Lambda})\boldsymbol{\theta}(t) + \mathbf{\Lambda}(\mathbf{W}_{sys}\boldsymbol{\theta}(t)) + \mathbf{W}_{in}\mathbf{v}(t), \quad (1)$$

$$\mathbf{y}(t) = \mathbf{W}_{out}\mathbf{x}(t), \quad (2)$$

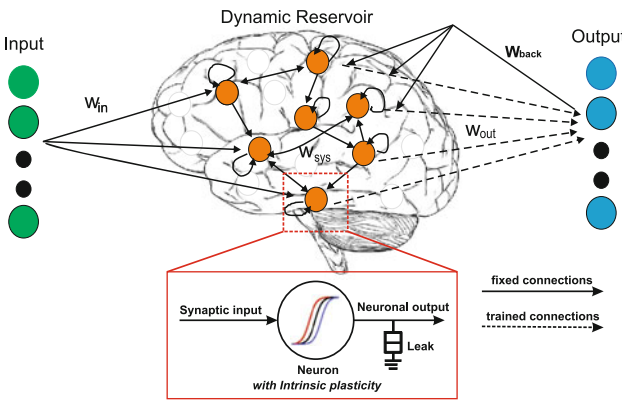



**Fig. 1** The Reservoir network architecture, showing the flow of information from input to reservoir to output units. Typically only the output connections $W_{out}$ are trained. The input connections $W_{in}$ and internal connections $W_{sys}$ are set randomly. Feedback connections $W_{back}$ from the output to the reservoir neurons if provided, are typically also set randomly. The highlighted section shows a zoomed in view of a single reservoir neuron (only a subset of the reservoir neurons depicted for the purpose of illustration)

[1] NARMA-30 is the 30th order non-linear auto-regressive moving average.

$$\lambda_i = \frac{1}{T_c}\left(\frac{1}{1+\rho_i}\right), \tag{3}$$

where $\mathbf{x}(t) = (x_1(t), x_2(t),\ldots, x_N(t))^T$ is the $N$ dimensional vector of dynamic reservoir state activation, $\mathbf{v}(t) = (v_1(t), v_2(t),\ldots, v_K(t))^T$ is the $K$ dimensional time dependent input that drives this recurrent network and $\mathbf{y}(t) = (y_1(t), y_2(t),\ldots, y_L(t))^T$ is the $L$ dimensional vector of output neurons. Each reservoir neuron has its own leak-decay rate $\lambda_i$ where $\mathbf{\Lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_N)^T$ is the collection of these individual leak decay rates. These leak values are inversely proportional to a leak control parameter, $\rho_i \in \{0, 1, 2, \ldots, 9\}$ and modulated by a global time constant $T_c > 0$. Most models use an uniform leak-rate or manually adjust this to a fixed value. However here they are determined by the local active information storage based adaptation rule (Sect. 2.3). The firing rate of each reservoir neuron is given by the vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_N)^T$, where

$$\theta_i(t) = \tanh(a_i x_i(t) + b_i). \tag{4}$$

Here $b_i$ acts as the individual neuron bias value, while $a_i$ governs the slope of the firing rate curve. We adapt these parameters according to a stochastic learning rule based on a generalized intrinsic plasticity mechanism, presented in Sect. 2.2.

The output weights $\mathbf{W}_{out}$ (Eq. 2) can be computed as the linear regression weights of the teacher outputs $\mathbf{d}(t)$ on the reservoir states $\mathbf{x}(t)$. The basic objective of such supervised training is to find a set of output weights such that the summed squared error between the desired output and the actual network output $\mathbf{y}(t)$ is minimized by changing the weights incrementally in the direction of the error gradient. One way to do this is by calculating the output weights $\mathbf{W}_{out}$ using the collection of the desired output states $\mathbf{D}$, and the pseudo-inverse of the matrix $\mathbf{S}$ collecting the states of the reservoir over a number of time steps as $\mathbf{W}_{out} = \mathbf{S}^{+}\mathbf{D}$ (*off-line training*). We use an alternative approach (*online training*) with no internal reservoir states being collected. Using the recursive least squared algorithm (RLS) (Jaeger 2003), we adapt the output weights at each time step. While the training inputs $\mathbf{v}(t)$ are being fed into the dynamic reservoir. We implement the RLS algorithm using a fixed forgetting factor ($\lambda_{RLS} < 1$). However as demonstrated in (Paleologu et al. 2008), it is possible to use an adaptive forgetting factor with an additional error change detection module.

RLS algorithm for self-adaptive reservoir training:

*Initialize:* $\mathbf{W}_{out} = 0$, exponential forgetting factor ($\lambda_{RLS}$) is set to a value close to 1 and the auto-correlation matrix $\rho$ is initialized as $\rho(0) = \mathbf{I}/\delta$, where $\mathbf{I}$ is unit matrix and $\delta$ is a small constant.

*Repeat:* At time step $t$

Step 1: For each input signal $\mathbf{v}(t)$, the reservoir state $\mathbf{x}(t)$ and network output $\mathbf{y}(t)$ are calculated using Eqs. (1) and (2).

Step 2: Training error $e(t)$ calculated as: $e(t) \leftarrow d(t) - \mathbf{W}_{out}(t-1)\mathbf{x}(t)$.

Step 3: Gain vector $\mathbf{K}(t)$ is updated as: $\mathbf{K}(t) \leftarrow \frac{\rho(t-1)\mathbf{x}(t)}{\lambda_{RLS} + \mathbf{x}^T(t)\rho(t-1)\mathbf{x}(t)}$.

Step 4: Update the auto-correlation matrix $\boldsymbol{\rho}(t)$: $\boldsymbol{\rho}(t) \leftarrow \frac{1}{\lambda_{RLS}}\left[\boldsymbol{\rho}(t-1) - K(t)\mathbf{x}^T(t)\boldsymbol{\rho}(t-1)\right]$.

Step 5: Update the instantaneous output weights $\mathbf{W}_{out}(t)$: $\mathbf{W}_{out}(t) \leftarrow \mathbf{W}_{out}(t-1) + K(t)e(t)$.

Step 6: $t \leftarrow t+1$. *Until:* Maximum number of time steps is reached

In order to further eleborate the working of the online RLS learning, we take the example of a Sine wave transformation function inspired from (Jaeger 2003). Starting with a moderate network size of $N = 50$ reservoir neurons, the inner network connections ($\mathbf{W}_{sys}$) were scaled to a spectral radius of 0.95 and the RLS learning rate was fixed at $\lambda_{RLS} = 0.998$. For the auto-correlation matrix initialization, we set $\delta = 10^{-5}$. All other network parameters were fixed equal to the ones used by the remaining experiments (Sect. 3.1). The input to the network is a sinosoidal signal of the form $v(t) = sin(\frac{\pi}{2} + 5\pi)$ (see green line Fig. 2a). Post training the network learns the output weights from the inner reservoir neurons to a single output neuron, such that it can produce a transformed sinosoidal signal of the form $d(t) = \frac{1}{2}sin^9(\frac{\pi}{2})$ (see blue line in Fig. 2a). In order to make the system robust to perturbations, white noise with zero mean and standard deviation 0.001 was added to all the reservoir neurons. This setup was run for a total of 3000 time steps with the first 50 time steps used as washout period, with the RLS learning switched off. After this, the RLS learning was switched on for the next 500 time steps. The network was then allowed to generate the desired output signal with the learnt output weights $\mathbf{W}_{out}$. The task is easily learned by the reservoir in the first 500 time steps as observed in Fig. 2a. Here the first 500 time steps were used for teacher forcing with RLS output weight learning. After this period, learning was stopped and the learned weights were used to generate the desired output signal. We randomly select four reservoir neurons in order to display the convergence of the learned output weights. From Fig. 2b it is observed that after the first 500 time steps the output weights converge resulting in the network generating the desired transformation along with the incoming input signal, without further teacher forcing.

## 2.2 Generic intrinsic plasticity

Homeostatic regulation by way of intrinsic plasticity (IP) is viewed as a mechanism for the biological neuron to modify its firing activity to match the input stimulus distribution (Turrigiano et al. 1994; Desai et al. 1999). In (Triesch

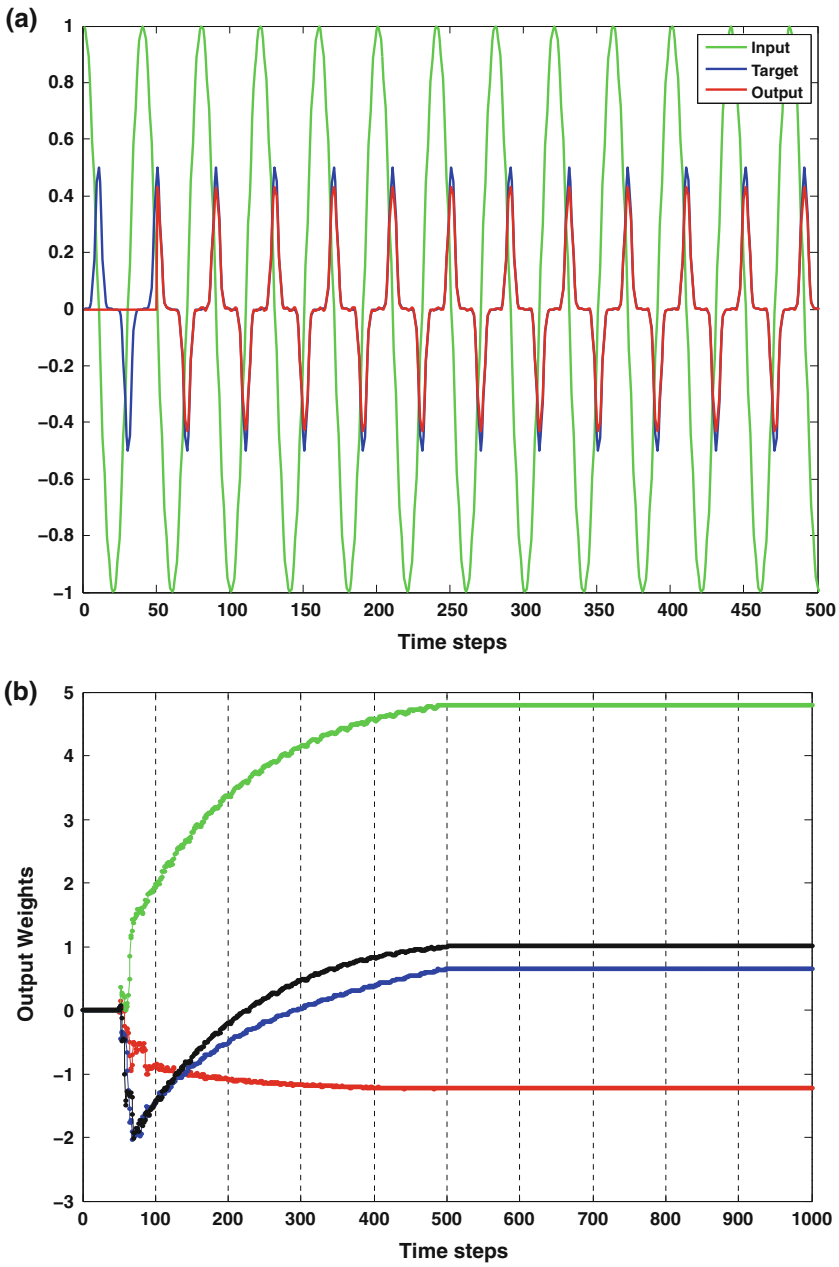



**Fig. 2** **a** Graph of the input signal $sin(\frac{\pi}{2} + 5\pi)$ (*green*) plotted against the desired target signal $\frac{1}{2}sin^9(\frac{\pi}{2})$ (*blue*) and the actual learned output (*red*) from the reservoir network. **b** The RLS algorithm learnt weight convergence of four randomly selected reservoir neurons

2007) a model of intrinsic plasticity based on changes to the neuronal non-linear activation function was introduced. A gradient rule for direct minimization of the Kullback–Leibler divergence between the neuronal current firing-rate distribution and maximum entropy (fixed mean) exponential output distribution was presented. Subsequently in (Schrauwen et al. 2008) an IP rule for the hyperbolic tangent transfer function with a Gaussian output distribution (fixed variance maximum entropy distribution) was derived. During testing the adapted reservoir dynamics, it was observed that for tasks requiring linear responses (e.g NARMA) the Gaussian distribution performs best. However on non-linear tasks, the exponential distribution gave a better performance. In this work, with the aim to obtain sparser output codes with increased signal to noise ratio for a stable temporal memory task, we implement the learning rule for IP using a Weibull output distribution as the target distribution.

The Weibull distribution (Eq. 7) is a two parameter continuous distribution where its shape parameter ($\alpha$) can be tweaked to generate a wide family of other popularly used probability distributions. As such with appropriate parameter choice, it can account for various shapes of the neuron transfer function (Eq. 4). With the aim for a high kurtosis number (sparser output codes) and generalization to different neuron activation functions, we choose the

shape parameter $\alpha = 3.5$. This model was very recently introduced in (Li 2011). However, the application of this rule in the reservoir computing framework and its effect on the network performance for standard benchmark tasks had not been studied so far. Furthermore, in contrast to the original model, we extend this for the tan-hyperbolic (tanh) neuronal nonlinearity.

The probability distribution of the two-parameter Weibull random variable θ is given as follows:

$$f_{weib}(\theta;\beta,\alpha) = \begin{cases} \frac{\alpha}{\beta}\left(\frac{\theta}{\beta}\right)^{\alpha-1} exp - \left(\frac{\theta}{\beta}\right)^{\alpha} & \text{if } \theta \geq 0 \\ 0 & \text{if } \theta < 0 \end{cases} \quad (5)$$

The parameters $\alpha > 0$ and $\beta > 0$ control the shape and scale of the distribution respectively. Between $\alpha = 1$ and $\alpha = 2$, the Weibull distribution interpolates between the exponential distribution and the Rayleigh distribution. Specifically for $\alpha = 5$, we obtain an almost normal distribution. Due to this generalization capability it serves best to model the actual firing rate distribution and also account for different types of neuron non-linearities. The neuron firing rate parameters $a$ and $b$ of Eq. (4) are calculated by minimizing the Kullback–Leibler divergence between the real output distribution $f_\theta$ and the desired distribution $f_{weib}$ with a fixed mean firing rate $\beta = 0.3$. Here the Kullback–Leibler divergence is given by:

$$\begin{aligned} D_{KL}(f_\theta, f_{weib}) &= \int f_\theta(\theta) log\left(\frac{f_\theta(\theta)}{f_{weib}(\theta)}\right) \mathrm{d}\theta \\ &= -H(\theta) + \frac{1}{\beta^\alpha} E(\theta^\alpha) - (\alpha - 1) E(log(\theta)) \\ &\quad - log\left(\frac{\alpha}{\beta^\alpha}\right) \end{aligned} \quad (6)$$

where, $f_\theta(\theta) = f_x(x)/\frac{\partial\theta}{\partial x}$. This is for a single neuron with input x and output θ. $H(\theta) = \int f_\theta(\theta) log f_\theta(\theta) \mathrm{d}\theta$ is the entropy and $E$ represents the expectation values.

Differentiating $D_{KL}$ with respect to $a$ and $b$ (see "Appendix" for details) we get the resulting online stochastic gradient descent rule for calculating $a$ and $b$ with the learning rate $\eta$ at each time step as:

$$\Delta b = -\eta\left[2\theta + \theta^{-1}(1-\theta^2)\left(\frac{\alpha}{\beta^\alpha}\theta^\alpha - \alpha + 1\right)\right]. \quad (7)$$

$$\Delta a = \frac{\eta}{a} + x\Delta b \quad (8)$$

In general this type of intrinsic plasticity tries to optimize the neuronal information content with respect to the incoming input signal. By contrast, the neural local memory adaptation rule (Sect. 2.3) tries to modulate the neuronal leakage. This is based on a quantification of the extent of influence that the past activity of a neuron has on it's activity in the next time step (immediate future). Therefore we combine IP learning with the neuron memory adaptation rule in series, such that the leakage adaptation is carried out after the intrinsic adaptation of the neuron non-linearity. This combination leads to a single self-adaptive framework that controls the local memory of each neuron based on the incoming input to the network, while preventing runway dynamics (homeostasis).

### 2.3 Neuron memory adaptation: information storage

In case of neurons with a certain degree of leakage (applied after the non-linearity) as introduced first in (Jaeger et al. 2007) for the leaky echo-state networks variant of reservoirs, the leakage rate $\lambda$ (see Fig. 1) determines how much a single neuron depends on the actual net input it receives, as compared to the influence of its own previous activity. Since $\lambda$ varies between 0 and 1, $1 - \lambda$ can be viewed as a local neuron memory term. The lower the value of $\lambda$, the stronger the influence of the previous level of activation as compared to the actual current input to the neuron. Hence if $\lambda = 1$, the neuron's previous activation has no effect on its present behavior or in other words the neuron has zero internal memory.

In order to account for an adjustable neuronal leak rate as a model of the leak in cellular membranes that works in conjunction with the IP rule, we use the local active information storage measure at each internal neuronal state. *Active information storage* (see Fig. 3a) introduced by (Lizier et al. 2012) refers to the amount of information in the previous state of the neuron that is relevant in predicting its immediate future state. It measures the amount of information stored in the current state of the neuron, that provides either positive or negative information towards its next state. Specifically, the instantaneous information storage for a variable $x$ is the local (or un-averaged) mutual information between its semi-infinite past $x_t^{(k)} = \{x_{t-k+1},\ldots, x_{t-1}, x_t\}$ and its next state $x_{t+1}$ at the time step $t + 1$ calculated for finite-$k$ estimations. Hence, the local information storage is defined for every spatio-temporal point within the network (dynamic reservoir). The local unaveraged information storage can take both positive as well as negative values, while the active (average) information storage $A_x(k) = \langle a_x(i,t,k)\rangle_t$ is always positive and bounded by the average information capacity of a single neuron state. Interestingly another information theoretic quantity, namely excess entropy, also provides a measure of the stored information. However it estimates the stored information which will be used at some arbitrary point in the future and not necessarily be the next time step $t + 1$ (Lizier et al. 2011). It is due to this reason the local active information storage serves as a more suitable measure of the neuron local memory.

The local information storage for an internal neuron state $x_i$ is given by:

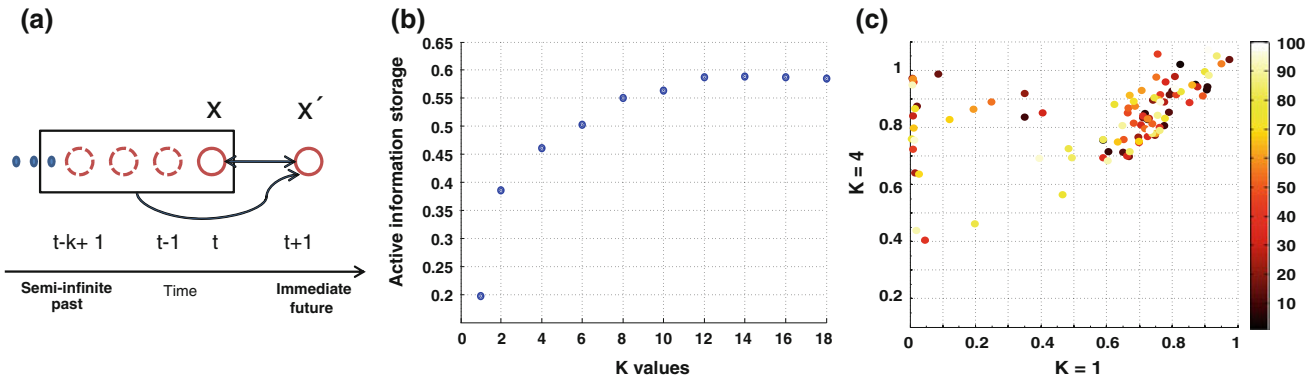



**Fig. 3** Pictorial representation of active information storage (AIS) calculated for a single neuron state $x$ and its immediate future state $x'$ (*solid circle* present and next time step states of the neuron, *dotted circle* previous states of the same neuron). **b** Active information storage convergence: plot of estimated AIS versus the history length $k$. **c** Plot of the change in local active information storage values (unaveraged) for 100 neurons with baseline history length $k = 1$ versus $k = 4$. Here, a concentration of points in the *upper left corner* of the graph, clearly depicts higher local AIS values for increasing $k$ value, as compared to the baseline estimate. Typically some neurons capture much higher information storage values as compared to others, due to their difference in activation (*colormap* represents the different neurons (1–100) (color figure online)

$$a_x(i, t+1) = \lim_{k\to\infty} \log_2 \left( \frac{P(x_{i,t}^{(k)}, x_{i,t+1})}{P(x_{i,t}^{(k)}) P(x_{i,t+1})} \right), \tag{9}$$

where $a_x(i, t + 1, k)$ represents finite-$k$ estimates and $A_x = \lim_{k\to\infty} \log_2 A_x(k)$. $k = 1$ is the natural starting choice for calculations of the estimates. However with increasing values of $k \to \infty$, the estimates tend towards the actual active information storage value, with a saturation point reached for a certain finite $k$-value. Beyond this point with an increase in $k$ there is no significant change in the finite-estimate of the information storage quantity (see Fig. 3b, c). Using epochs($\phi$) with finite history length $k = 8$, the active information storage measure at each neuron adapts the leak control parameter $\alpha_i$ as follows :

$$\rho_i = \begin{cases} \rho_i + 1 & \text{if } A_x(i, \phi) - A_x(i, \phi - 1) > \epsilon \\ \rho_i - 1 & \text{if } A_x(i, \phi) - A_x(i, \phi - 1) < \epsilon, \end{cases} \tag{10}$$

where $\epsilon = \frac{1}{4} \log_2 N$ and $0 < \alpha_i < 9$.

After each epoch, $\rho_i$ and $\lambda_i$ (Eq. 3) are adjusted and these values are used for the subsequent epoch. Once all training samples are exhausted, the pre-training of the reservoir is completed and $\lambda_i$ is fixed. The information storage measure was implemented using the Java based information dynamics toolkit (Lizier 2012). The toolkit was used as a wrapper class with Matlab.

Note that it can be observed from Eq. (1) that the reservoir time-scale is controlled by the leak($\mathbf{\Lambda}$) matrix. Thus, the adaptation of the individual leak-rates can be observed as the tuning of neuronal time-constants. Since this is governed by the change in information storage of each neuron based on the incoming input, the reservoir speeds up or slows down its dynamics depending on the time scales of input signal.

## 3 Experimental setup and analysis

The performance of our self-adaptive reservoir network on delay temporal memory tasks is evaluated first by testing it on two benchmark tests, namely the NARMA-30 time series modeling task, and a delayed 3-bit parity task. These have been used as standard for comparison of memory performance by the reservoir computing community. By taking the inherent non-linearity and the requirements for a extended temporal memory into account, both of these are complex signal processing tasks. In the second part of our experiments we use a classic delay temporal memory scenario of robot navigation through a T-shaped maze. This is evaluated for a simple simulated wheeled robot as well as a complex physical walking machine AMOS II. By generalizing between both small and long mazes, this task clearly demonstrates the potential application of our network for solving real robotic tasks with variable delay period between memory storage and retrieval.

### 3.1 Experimental setup

In all experiments here, the internal reservoir weights $\mathbf{W}_{sys}$ were drawn from an uniform distribution over $[-1, 1]$ and were subsequently scaled to a spectral radius of 1.2 (note that, intrinsic plasticity allows a spectral radius greater than unity and hence the reservoir network contains a wide spread distribution of neural signals). Input weights and output feedback weights (if provided) can be randomly generated in general. Here they were drawn from an uniform distribution over $[-0.5, 0.5]$. The firing rate parameters were initialized as $a = 1$ and $b = 0$. The learning rate for the stochastic gradient descent algorithm was fixed at $\eta = 0.0008$. Weibull IP and individual neuron leak

adaptation were carried out in 10 epochs of 1000 time steps, in order to determine the optimal parameters $a_i$, $b_i$ and $\lambda_i$ for each neuron. Performance evaluation was done after the neuron leak and transfer function parameters had been fixed. For all the standard benchmark tests the internal reservoir network was constructed using $N = 200$ leaky integrator neurons initialized with a 10 % sparse connectivity.

#### 3.1.1 Dynamic system modeling with 30th order NARMA

The dynamics of the *nth* order non-linear auto-regressive moving average is given by:

$$z(t+1) = 0.2z(t) + 0.004z(t)\sum_{i=0}^{n-1} z(t-i) + 1.5v(t-(n-1))v(t) + 0.001 \quad (11)$$

Here $n = 30$ for the 30th order modeling scenario and $z(t)$ is the output of the system at time 't'. $v(t)$ acts as the input to the system at time 't', and is uniformly drawn from the interval [0,0.5]. The task is to output $z(t)$ based on $v(t)$. In general this task is quite complex considering that the current system output depends on both the current time step input as well as its own previous $n - 1$ time steps history. Consequently, we use feedback connections ($\mathbf{W}_{back}$) from the output neurons to the internal neurons with Eqs. (1) and (2) modified to:

$$\mathbf{x}(t+1) = (\mathbf{I} - \mathbf{\Lambda})\boldsymbol{\theta}(t) + \mathbf{\Lambda}(\mathbf{W}_{sys}\boldsymbol{\theta}(t)) + \mathbf{W}_{in}\mathbf{v}(t) + \mathbf{W}_{back}\mathbf{y}(t) \quad (12)$$

$$\mathbf{y}(t+1) = \mathbf{W}_{out}[\mathbf{x}(t), \mathbf{y}(t)] \quad (13)$$

The main goal of the NARMA task is to evaluate the ability of the reservoir to model a highly non-linear system where the system state depends on the incoming input as well as its own history. Due to this inherent dependence on its own previous history this task requires extended temporal memory with increasing complexity for higher orders of the system. The training, validation and testing were carried out using 1,000, 2,000 and 3,000 time steps respectively. Five fold cross-validation was used with the training set. Here the first 50 steps were used to warm up the reservoir and were not considered for the training error measure. The network setup consisted of a single input neuron, feeding the input $v(t)$ to the reservoir network and just one output neuron. We evaluated the network performance in this task using the normalized root mean squared error between the desired signal $d(t)$ and the actual network output signal $y(t)$:

$$NRMSE = \left(\frac{\langle (d(t) - y(t))^2 \rangle}{\langle (d(t) - \langle y(t) \rangle)^2 \rangle}\right)^{\frac{1}{2}} \quad (14)$$

#### 3.1.2 Delayed n-bit parity task

The delayed $n$-bit parity task functions over input sequences $t$ time steps long, and determines for $n$ bits, if $\tau + n \rightarrow \tau$ time steps in the past are active. Here $\tau$ represents the delay period. The input consists of a temporal signal $v(t)$ drawn uniformly from the interval $[-0.5, 0.5]$. Using $n = 3$ bits, the desired output signal is calculated as the PARITY check $(v(t-\tau) \oplus v(t-\tau-1) \oplus v(t-\tau-2)$ for increasing time delays of $0 \leq \tau \leq 400$. Since the parity function (XOR) is not linearly separable, this task is quite complex and requires the ability to recall long spans of memory. The network setup consisted of a single input neuron, the internal reservoir network with 200 neurons and 400 output units. We evaluated the memory capacity of the network as the amount of variance of the delayed input signal recoverable from the optimally trained output units summed over all delays. This measure was first introduced by (Jaeger 2001). For a given input signal delayed by $k$ time steps, the net memory capacity is given by:

$$MC = \sum_k MC_k = \sum_k \frac{cov^2(y(t-k), d(t))}{var(y(t))var(d(t))} \quad (15)$$

where cov and var denote covariance and variance operations, and as before $y(t)$ and $d(t)$ represent the desired and actual output signals.

#### 3.1.3 Robot T-maze navigation

In order to demonstrate the temporal memory capacity of our system, we employ a variable delay temporal memory task of navigation through a T-shaped maze. The experiments are carried out first in simulation using a simple wheeled robot NIMM4 (Fig. 4) and then finally with a more complex physical walking robot AMOS-II (Fig. 5). In case of the simulation task a reservoir size of $N = 200$ neurons was used, while the reservoir size for the real robot experiment was considerably larger with $N = 500$ neurons. This was fixed, keeping in mind the extended delay memory required for the T-maze in the real robot experiments. However, in both cases the reservoir has 10 % sparse connectivity. The simulated robotic task was performed using 6 input neurons (number of sensors) and 2 output neurons (number of actuators). In case of the real robot experiment, we use 4 input neurons and 2 output neurons with the reservoir network (see Fig. 7).

The primary objective of this task is to let the robots move from the starting position until the end of the maze while making the correct turn at a recall zone (see Figs. 6a, b, 7c). While walking along the corridor, the robot receives a *cue* signal (a bright light activation in case of AMOS-II or the presence of a spherical object in case of simulation)

either to their left or right side. This provides information to the robots regarding the required turning behavior at the T-junction. On reaching the end of the corridor the robots should make the correct turn depending on this previously applied cue signal.,

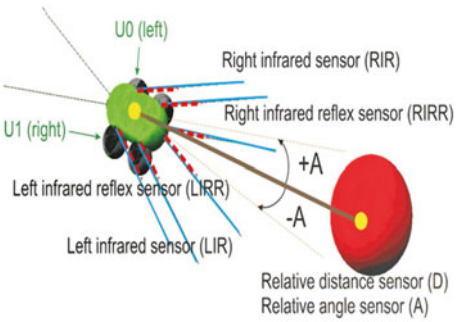



**Fig. 4** Model of the simulated wheeled robot NIMM4 showing the sensors (LIR, RIR, LIRR, RIRR) and actuators (U0,U1). The *red ball* in front of the robot represents its goal

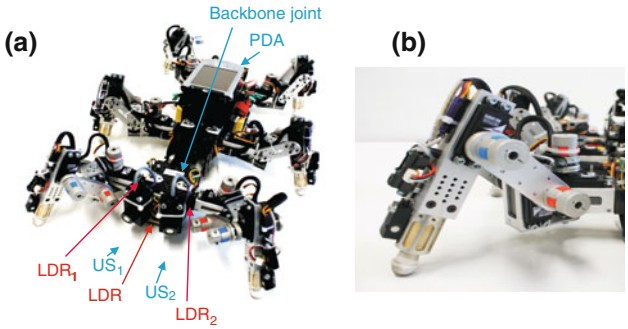



**Fig. 5** **a** Biologically inspired six-legged walking machine AMOS II. **b** Leg structure of AMOS II inspired from a cockroach leg (showing the three different leg joints) (color figure online)

In order to demonstrate the generalization capability of the system to longer time delays, we divided the task into two mazes (see Fig. 6a) of different lengths. Maze B requires a longer temporal memory (larger delay between cue and recall) as compared to maze A. Furthermore, in case of the simulated wheeled robot, we had a more controlled environment with a much smaller delay timescale (seconds) while in the hexapod robot, the actual maze is considerably big with larger delay timescale (minutes). Here the robot has to learn both the reactive behavioral task of turning at the T-junction as well as remembering the cue signal shown much before, to negotiate the correct turn. As such conventional methods, like landmarks to identify the T-junction, are not needed.

*Complex physical walking robot AMOS II*: AMOS II (successor to AMOS robot (Steingrube et al. 2010)) is a biologically inspired hardware platform (Fig. 5) having six identical legs similar to an insect. Each leg has three joints. The morphology of these multi-jointed legs is modeled on the basis of a cockroach leg but with the tarsus segments ignored. The body of AMOS II consists of two segments: a front segment where two forelegs are installed and a central body segment where the two middle and the two hind legs are attached. They are connected by one active backbone joint inspired by the invertebrate morphology of the American cockroach's trunk. This backbone joint is for up- and downward bending, which allows it to climb over obstacles. All leg joints including the backbone joint are driven by digital servomotors.

The size of AMOS II is 30 cm wide, 40 cm long, 22 cm high. The weight of the fully equipped robot (including 19 servomotors, all electronic components, sensors, and a mobile processor) is approximately 4.5 kg. AMOS II has a

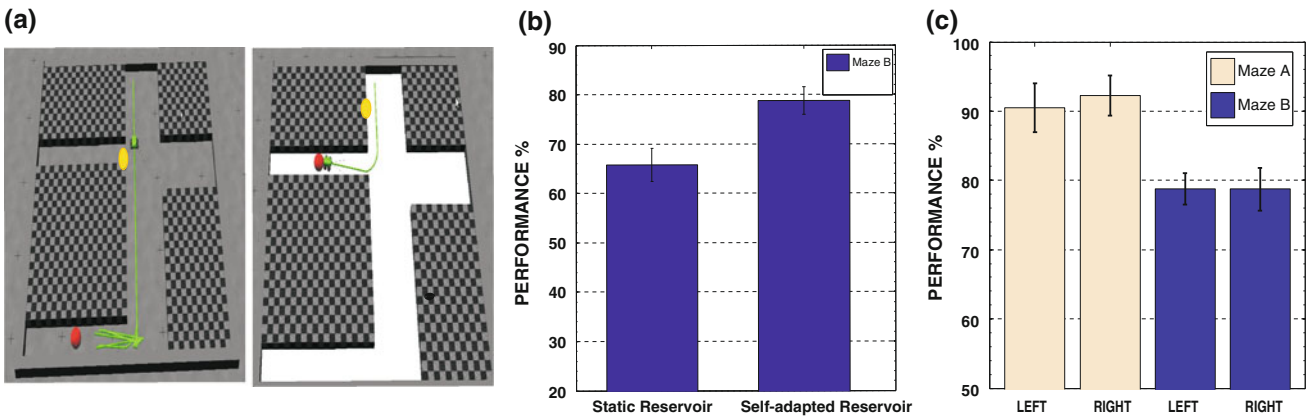



**Fig. 6** **a** Screenshots of the robot successfully navigating through the long maze B (*left*) and the short maze A (*right*). *Yellow ball* is cue to turn *right* at the T junction, *red ball* marks delay time between cue and the recall zone. **b** Performance on the large maze B simulation task after 80 trials for static reservoir versus our self-adapted reservoir. Our network outperforms by 10 %. **c** Performance of the robot in the simulation task with the two mazes (maze A shorter than maze B) measured in terms of the percentage of correct times the robot took the proper trajectory (*left*/*right* turn at the T-junction) to reach the end point. 5 % noise is considered on all sensors (color figure online)

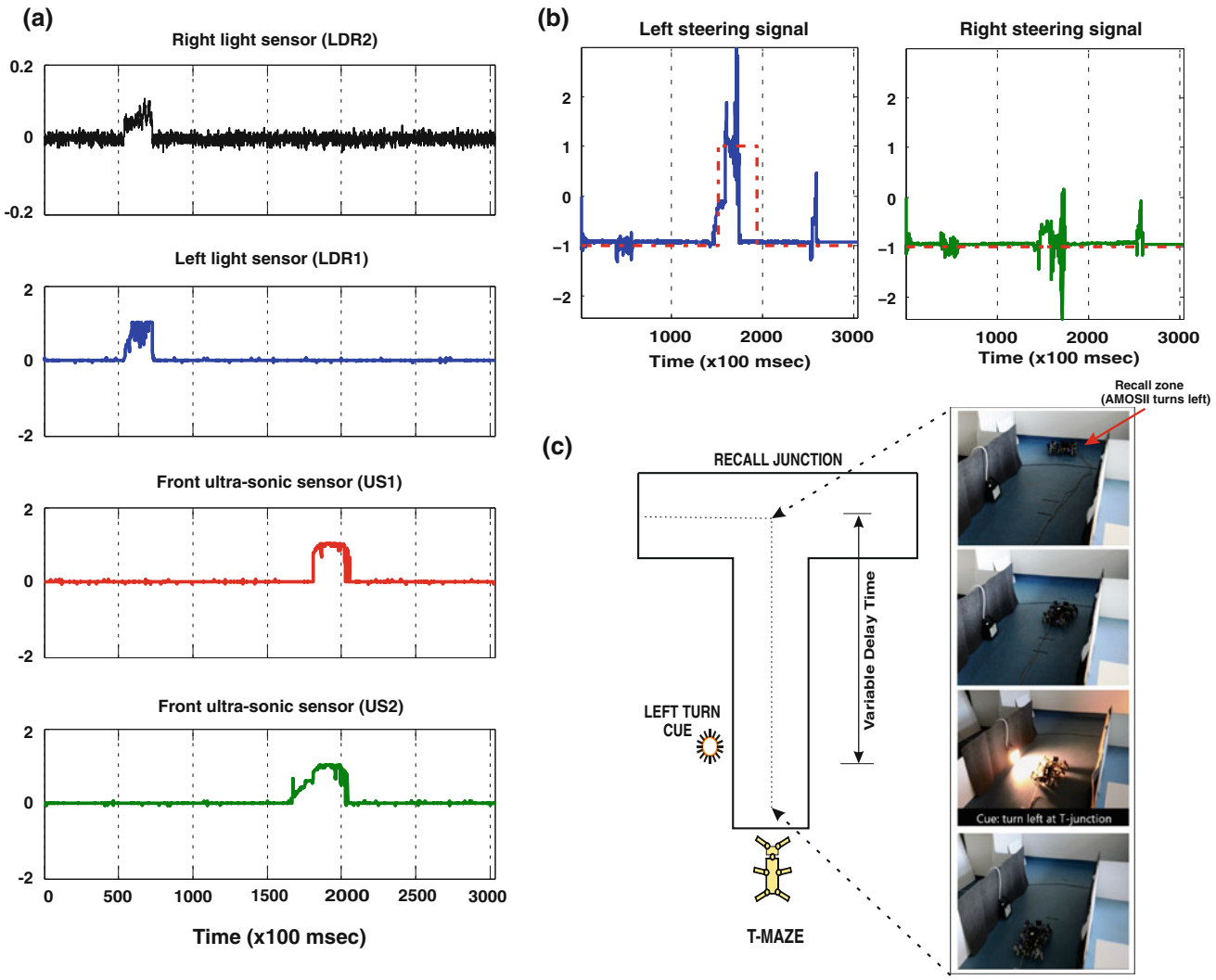



**Fig. 7** **a** Plots of the sensor signals from AMOS II recorded during the experiment, which act as the four inputs to the reservoir network. The signals shown are from a single run where the cue signal (light source) was applied to the *left*, while walking along the corridor (LDR1 $\gg$ LDR2). The two ultrasonic sensors become active at the same time when AMOS-II reaches the T-junction (cue recall zone). **b** Plots the trained reservoir network outputs (*solid-line* learned behavior, *dotted-line* desired behavior). Here the *left* steering signal is active (+1) while the *right* steering signal is inactive (−1) and the robot makes a *left* turn (behavior learned at the same time step of the activation of the US1 and US2 sensors indicating the recall zone. **c** Pictorial representation of the T-shaped maze setup. While walking along the long corridor, a cue in the form of a light signal is applied either to the *left* or *right side* of AMOS II. The robot needs to recall this cue at the *recall junction* and execute the corresponding turning behavior. The temporal delay between the time of presentation of *cue* and the end of the corridor (T-junction) is the total memory span. This can vary with different delay times for small and long mazes. The screenshots (*right*) from the experiment show the actual behavior of the hexapod while walking along the corridor

total of 17 sensors. For the maze-navigation experiments we only make use of the two light dependent resistor sensors ($LDR_{1,2}$) on the left and right sides of the front body part, and the front two ultrasonic sensors ($US_{1,2}$). These act as the sensory inputs to the reservoir network for the T-maze navigation task. We use a Multi-Servo IO-Board (MBoard) installed inside the body to digitize all sensory input signals and to generate a pulse-width-modulated signal to control servomotor position. The MBoard is connected to a personal computer (PC) via an RS232 interface. Electrical power supply is provided by batteries: one 11.1 V lithium polymer 2,200 mAh for all servomotors, two 7.4 V lithium polymer for the electronic board (MBoard) and for all sensors. For more information of AMOSII, please refer to (Ren et al. 2012; Manoonpong et al. 2013b).

The experiment consisted of three parts. In the first part dataset acquisition was done using human controlled navigation of AMOS II through the maze and the sensor and steering signal (see Fig. 7a, b) readings were recorded. Twenty runs with different starting positions for both, left and right turn cues were carried out. This was done for

both, small and long time delays, between cue and recall zone. This data was then used for the training of the reservoir network. Finally online testing was carried out with the trained steering signals being fed into the AMOS II controller.

*Simulated wheeled robot NIMM4*: The simulation robot NIMM4 consists of four infrared sensors (LIR, LIRR, RIR, RIRR), a relative distance sensor (D), a relative angle of deviation sensor (A) and four actuators to control the desired turning and speed. The experiment consists of dataset acquisition, training of our adapted RC and off-line testing. During the first phase using the simulator, we manually controlled the robot movement through the maze using simple keyboard instructions and recorded the sensor and actuator values. We recorded 80 examples in total with different initial starting positions. 40 % of these were used for training and 60 % for testing purposes. After the first phase, the self-adapted RC was trained using imitation learning on the collected data with the actuator values from manual control as desired output. Finally we performed off-line testing using the remaining set of recorded data. Simulations were carried out using the C++ based LPZ-Robot simulator.[2]

## 4 Results

In Table 1. we summarize the standard benchmark tests results of our self-adaptive reservoir network in comparison to the performance obtained by a static RC and RC with only Gaussian distribution based intrinsic plasticity (Schrauwen et al. 2008). All the parameters for the compared RC's were set to their critical values, such that they operated at their optimal regime of performance (Bertschinger and Natschläger 2004). Our network clearly outperforms the other two networks, both in terms of lowest normalized root mean squared error (0.362) for the 30th order NARMA task, as well as an extended average memory capacity of 47.173 for the delayed 3-bit parity task. Non-normal networks (e.g. a simple delay line network) have been shown to theoretically allow extensive memory (Ganguli et al. 2008) which is arguably not possible for arbitrary recurrent networks. However our self-adaptive RC network shows considerable increase in the memory capacity (with 400 reservoir neurons), which was previously shown to improve only in case of specifically selected network connections (permutation matrices as internal network weight configurations) (Boedecker et al. 2009).

[2] It is based on the Open Dynamics Engine (ODE). More details of the LPZRobot simulator can be found at http://robot.informatik.uni-leipzig.de/software/.

**Table 1** Normalized root mean squared error (NRMSE) and average memory capacity performance for the NARMA-30 and 3-bit parity tasks, comparing the basic RC (ESN) model, the RC model with a intrinsic plasticity method using Gaussian probability density and our self-adapted RC (SRC) network using Weibull probability density (optimal values in italics)

| Dataset | Measure | RC (ESN) | IP (GAUSS) | SRC |
|---|---|---|---|---|
| NARMA-30 | NRMSE | 0.484 | 0.453 | *0.362* |
| | SD | 0.043 | 0.067 | 0.037 |
| 3-bit Parity | MC | 30.362 | 32.271 | *47.173* |
| | SD | 1.793 | 1.282 | 1.831 |

We further test the delay memory capability of the self-adapted reservoir with the robot maze navigation tasks. In Fig. 6a we show screenshots of the simulated robot performing the maze navigation task and successfully making the correct turn at the T-junction for both long (left image) and short (right image) mazes. The turn depends on the prior input appearing while driving along the corridor. The robot NIMM4 can have different speeds while moving through the maze. In general the robot has a faster speed in the corridor and a comparatively slower speed while negotiating turns. Our network with the leak adaptation method can easily deal with this situation and as such successfully learns this task. It only uses the sensor data to drive along the corridor and outputs the desired actuator values to move along the correct trajectory while turning at the T-junction. The off-line testing results in the form of the percentage of correct turns from the total test set for both mazes are shown in Fig. 6c. In case of the shorter maze A (smaller delay between cue and recall) we achieve average performance of 92.25 % (±2.88 standard deviation). A good generalization capability for the longer maze B is also observed with the average performance of 78.75 % (±3.11 standard deviation), both for right turn. This is quite high as compared to previous results obtained by (Antonelo et al. 2008) for a similar task with a static Echo-state network. Furthermore in Fig. 6b one can see that the adapted reservoir network clearly outperforms a static RC for the same task by a margin >10 %. Here we compare the two reservoirs based on the performance only for the longer maze B, as this required a much larger delay memory capacity. The overall performance can be further enhanced if additional sensors were available to the robot, owing to the availability of additional information and more inputs to the reservoir network.

In comparison to the simulated task the maze navigation scenario with the physical robot AMOS II is more complex in terms of the much larger time scale of delay memory required. In simulation the largest maze B required a maximum of 50 time steps delay (time scale in seconds) between the cue and the recall, while the experiments with

AMOS II had a three times larger delay of 1,500 time steps (time scale × 100 ms) between cue and recall (Fig. 7a). Furthermore while the simulated task was performed in a controlled environment with the network tested off-line, the real robot experiments are carried out in an online setup. Note that AMOSII locomotion is driven by modular neural control (see Steingrube et al. 2010 for more details). Here the reservoir outputs are used to steer the robot by using the modular neural controller. In Fig. 7a we plot the sensor signals, that act as the input to the reservoir. The onset of $LDR_1$ triggers the left turn cue, while the simultaneous onset of both the front ultrasonic sensors $US_{1,2}$ signals at the recall zone. A high dimensional convolution of these signals reverberate as neural traces inside the reservoir network (a subset of these diverse set of signals is plotted in Fig. 8b). The local active information storage (Fig. 8a) for individual neurons shows that the two events of *cue* and *recall* are recognized as high information content regions (500 time step and 1,500 time step) while the neurons have a low local AIS value during the remaining time steps. This leads to the modulation of neuronal leak, with most neurons having a low leak (high local memory) at the time of left turn cue and then again at the end of the corridor (Fig. 7c) when recall signals get triggered. During the remaining time steps, the reservoir neurons have a higher leak-rate (low local memory). As the individual neuron leak-rates act as their local time scales and collectively control the timescale of the reservoir. This mechanism leads to a slowing down of the reservoir dynamics at high information content regions (cue and recall) and speeding up during the rest of the time. Using online learning, the reservoir network successfully learns the correct turning behavior. In this case due to the previously applied left turn cue, only the left steering signal is active while the right steering signal remains inactive and the robot makes a left turn. It is important that the robot starts turning at the correct time in order to prevent an early turn or crashing into the wall at the end of the corridor. This is clearly achieved as seen from the near perfect coincidence between the desired and actual outputs (Fig. 7b)[3].

The reservoir outputs are post-processed to get rid of signal noise before being feed into the modular neural controller of the robot. Averaging over 20 runs for both left and right turn scenarios, we achieved a performance of 80.23 %, for which the robot was able to successfully make the correct turn. In all cases the output signals were perfectly reconstructed. This performance was significantly higher as compared to a static reservoir network, which succeeded in making the correct turn on 62.54 % of cases. Without leak adaptation, in case of the static reservoir AMOS II showed a wall following behavior with turning being triggered much too early or the output signals reconstructed without threshold crossing (<1).

A suitable measure for the richness of the reservoir is believed to be the Average state entropy (ASE), with the instantaneous values showing how diverse the reservoir signals are in time. Moreover as mentioned in (Ozturk et al. 2007). ASE provides a measure for the volume of the reservoir manifold spanned by diverse signal trajectories. Using an approximation of the Renyi's quadratic entropy, the instantaneous state entropy for the reservoir states $\mathbf{x}(t) = x_1t, x_2t, \ldots, x_Nt$ can be calculated using a Gaussian kernel ($\sigma$) as follows:

$$H(x) = -log\left[\frac{1}{N^2}\sum_j\left(\sum_i \sigma(x_j - x_i)\right)\right] \tag{16}$$

Here we calculated the instantaneous state entropy values using a Gaussian kernel with radius 0.3. In Fig. 8c we clearly observe that the self-adaptive reservoir network achieves considerably higher average state entropy as compared to the static reservoir. Furthermore, as the spectral radius of the reservoir weight matrix increases there is a gradual increase in ASE values in both cases. However our network shows high ASE values for a spectral radius greater than unity, which is in sharp contrast to previous observations with static reservoirs like Echo state networks. This can be attributed to the intrinsic plasticity mechanism in the network, which allows for increased spectral spread of the network connectivity, with the higher ASE values indicating a much richer repertoire of activity within this network. In general keeping task independent performance in mind, it is desirable to have a large reservoir manifold volume. Here our adapted network clearly outperforms the static case.

## 5 Discussion and biological relevance

In this work we have presented and evaluated a self-adaptation mechanism for the reservoir computing network based on the information dynamics of the internal recurrent neural layer. This mechanism successfully combines an intrinsic plasticity rule using a generic probability distribution (Weibull) with a neuron leak adjustment rule based on local information storage measure. The neuron leak rate not only governs the degree of influence of local memory but also acts as the neuronal activity time-constant. Due to feedback connections in such recurrent networks, chaotic or runaway activity had been previously observed in the works of (Sompolinsky et al. 1988) and (Sussillo and

[3] The real robot experiment showing the cue signal activation and the corresponding turning behavior is demonstrated in a video clip at http://manoonpong.com/STM/AMOSII_stm.wmv

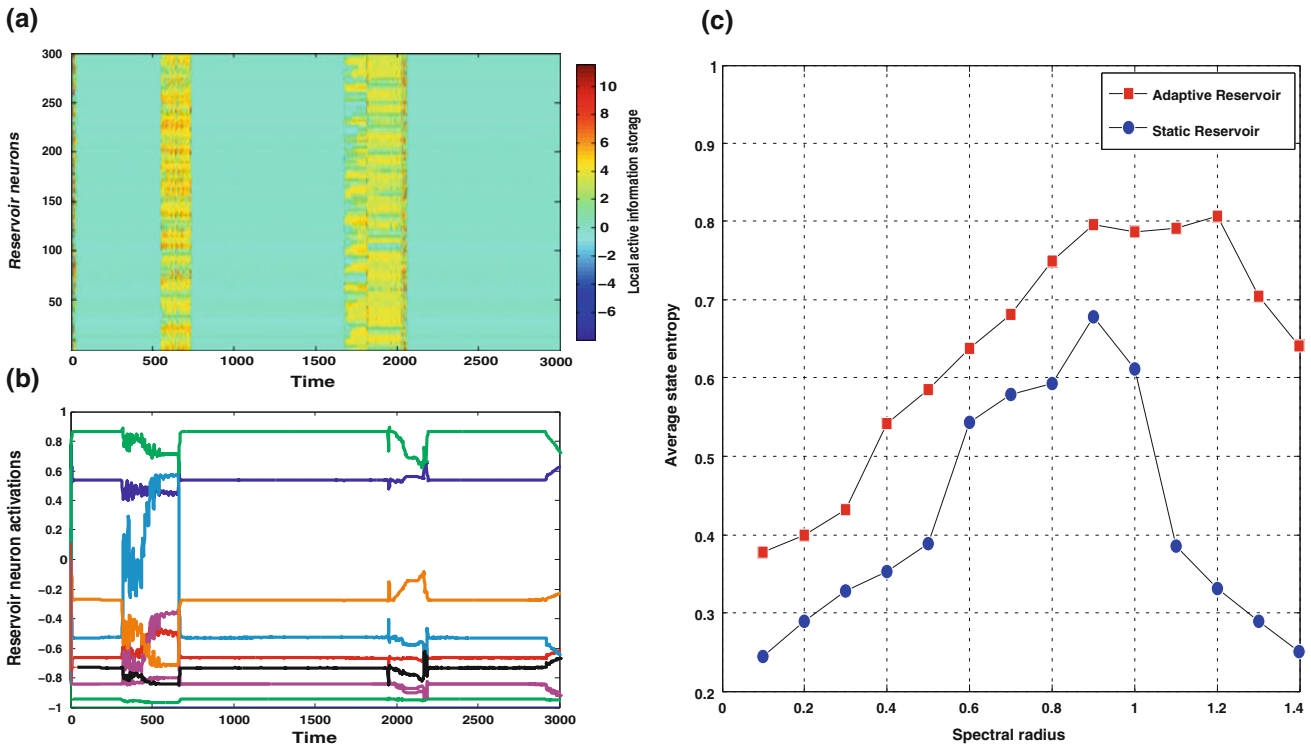



**Fig. 8** **a** Plot of the local active information storage values for 300 reservoir neurons at different time steps (*color coding* corresponds to the local active information storage values at different time steps). **b** Reservoir activations for a randomly selected subset of the neurons. **c** The plot of average state entropy for different spectral radius values of the reservoir connectivity matrix. Significantly higher entropy values observed for our adaptive network as compared to static reservoirs. Optimal spectral radius for static reservoir is between 0.9 and 1.0, while for the adaptive reservoir, optimal state entropy is reached at spectral radius of 1.2 (color figure online)

Abbott 2009). The intrinsic plasticity mechanism ensures information maximization at each neuron output while homeostatically regulating the network activity and prevents such runaway dynamics. In general, our mechanism allows minimal parameter tuning, with two of the important network parameters leak-rate, shape and scaling properties of neurons transfer function adjusted on the fly. In contrast most static reservoirs pre fix these parameter values or adapt them based on output error gradients that do not take into account difference in time scales of the input signal.

The ability to precisely track and tell time is critical for the learning of ordered motor behaviors as well as the underlying cognitive process, in all living creatures. However, the mechanism by which the brain tells time is still not clearly understood. Although it is still debated whether dedicated or intrinsic mechanisms underlie the timing process, some experimental and theoretical studies have validated the concept of neural circuits being inherently capable of sensing time across time scales (Tetzlaff et al. 2012). Large recurrent neural networks like these reservoir systems could be considered as an abstraction of the mammalian cortex. Accordingly (Buonomano and Laje 2010) suggested the concept of population clocks, where time is encoded in the time varying patterns of activity of neuronal populations, which emerge from the internal dynamics of the recurrent network. It is important to note that continuous input signals to these recurrent networks or the brain, in general can contain many different time scales. In order to account for varying time scales of input patterns to such networks, classically they have been setup in a hierarchical arrangement with different pre-determined time scales for each layer of hierarchy (Jaeger 2007; Yamashita and Tani 2008). However, monkey experiments (Bernacchia et al. 2011) have shown that individual neurons can have different time scales of reward memory correlated with the actual behavior. As such it is highly plausible that neurons in a single recurrent network can adjust or tune there individual time constants to account for a multi-time scale input in contrast to a hierarchical arrangement with different fixed time scales. As observed in Figs. 7a and 8a, high local active information storage regions in the network correspond to significant events in time. According to the learning rule from Eqs. (3) and (10) the individual neuron leak rates (time constants) have been adjusted according to the change of their AIS values with respect to a predefined threshold. In other words we were able to incorporate a self-adapting non-uniform leak rate in

the network that can account for varying time scales in the input stream as well as encode timing of events. As such in this work we not only present a mechanism to achieve a self-adaptive reservoir that can achieve a high degree of delayed memory capacity. From a biological perspective, we show that time is not only encoded in the internal recurrent dynamics but also single neurons may adjust their time-constants in order to account for high relevance events in the input data.

## 6 Conclusion

In this work we present a self-adaptive reservoir (RNN) framework such that using an information theoretic approach we have successfully adapted the local neuron (dynamic reservoir) time constants via it's leak rate, while at the same time the network maintains homeostasis through the generic intrinsic plasticity mechanism. The evaluated performance on the two standard benchmark tasks demonstrates that our adaptation mechanism clearly outperforms static reservoirs. Furthermore we demonstrate the application of our network to the control of autonomous robotic agents through the maze navigation experiments. Here our network is effective not only in reconstructing the original trajectories but can also cope with the variable temporal delay memory problem. It has been widely accepted that timing of events and memory guided behavior are intrinsically related. Specially for memory in the shorter time-scale of seconds to minutes (working memory Ungerleider et al. 1998), the system needs the ability to recognize important events in time. We achieve this in our network via the leak-adaption that allows the neurons to speed up or slow down their dynamics based on the incoming input, while at the same time encode highly relevant events using the active information storage measure. Both the simulation and real robot experiments demonstrate such memory guided behavior with the reservoir adapted according to the incoming sensory signals.

As a future direction more memory intensive tasks like simultaneous localization of multiple cue signals and cascading different temporal delays will be tested. Due to the universal computing power of recurrent neural networks, this type of adaptive reservoir can not only be used for temporal memory tasks, but also prove useful in generic signal processing requiring functional approximation of multi-time scale signals. Furthermore, we also aim to integrate our network with reinforcement learning techniques (Manoonpong et al. 2013a) requiring varying time scales of reward related memory for effective behavioral control of autonomous agents. Specifically on partially observable markov decision process (POMDP) problems which require a rich memory content for effective solution, our self-adaptive network offers potential applications.

**Acknowledgments** The research leading to these results has received funding from the Emmy Noether Program DFG, MA4464/3-1, by the European Communitys Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme3, Information and Communication Technologies) under grant agreement no.270273, Xperience, by the Federal Ministry of Education and Research(BMBF) by grants to the Bernstein Center for Computational Neuroscience (BCCN) Göttingen, grant number 01GQ1005A, project D1 and by the Max Planck Research School for Physics of Biological and Complex Systems.

## Appendix

The activation of each reservoir neuron with a `tanh` non-linearity with slope($a$) and shape($b$) parameters can be represented as $\theta = \texttt{tanh}(\texttt{ax} + \texttt{b})$. The activations are time dependent as shown in Eq. (4), however here we neglect the time variable for mathematical convenience. The tanh non-linearity can be represented in an exponential form as follows:

$$\theta = tanh(ax + b) = \frac{e^{2(ax+b)} - 1}{e^{2(ax+b)} + 1} \tag{17}$$

Differentiating this w.r.t $x$, $a$ and $b$ and representing in terms of θ we get the following set of base equations:

$$\begin{aligned} \frac{\partial\theta}{\partial x} &= a(1-\theta^2), \\ \frac{\partial\theta}{\partial a} &= x(1-\theta^2), \\ \frac{\partial\theta}{\partial b} &= (1-\theta^2) \end{aligned} \tag{18}$$

The probability distribution of the two-parameter Weibull random variable $\theta$ is given as follows:

$$f_{weib}(\theta;\beta,\alpha) = \begin{cases} \frac{\alpha}{\beta}\left(\frac{\theta}{\beta}\right)^{\alpha-1} exp - \left(\frac{\theta}{\beta}\right)^{\alpha} & \text{if } \theta \geq 0 \\ 0 & \text{if } \theta < 0 \end{cases} \tag{19}$$

Inorder to find a stochastic rule for the calculation of the neuron transfer functin parameters $a$ and $b$, we need to minimize the Kullback–Leibler (KL) divergence between the real output distribution $f_\theta$ and the desired distribution $f_{weib}$. The KL-divergence ($D_{KL}(f_\theta, f_{weib})$) is given by:

$$\begin{aligned} D = D_{KL}(f_\theta, f_{weib}) &= \int f_\theta(\theta) log\left(\frac{f_\theta(\theta)}{f_{weib}(\theta)}\right) \mathrm{d}\theta \\ &= \int f_\theta(\theta) log f_\theta(\theta) \mathrm{d}\theta - (\alpha - 1) \\ &\quad \times \int f_\theta(\theta) log(\theta) \mathrm{d}\theta \\ &\quad + \frac{1}{\beta^\alpha} \int f_\theta(\theta)\theta^\alpha \mathrm{d}\theta + \mathrm{C} \end{aligned} \tag{20}$$

Using the relation $f_\theta(\theta) = \frac{f_x(x)}{\frac{\partial\theta}{\partial x}}$ for a single neuron with input $x$ and output $\theta$ and representing the integrals in terms of the expectation(E) quantities, the above relation can be simplified to (here $C$ is a constant):

$$D = -E\left[log\left(\frac{\partial\theta}{\partial x}\right)\right] + E[logf_x(x)] + \frac{1}{\beta^\alpha}E(\theta^\alpha) - (\alpha-1)E(log(\theta)) + C \quad (21)$$

Using the partial derivatives from Eq. (18) and differentiating $D$ w.r.t the parameter $b$ yields:

$$\frac{\partial D}{\partial b} = E\left[2\theta + \frac{\alpha}{\beta^\alpha}\theta^{\alpha-1}(1-\theta^2) - (\alpha-1)\theta^{-1}(1-\theta^2)\right] = E\left[2\theta + \theta^{-1}(1-\theta^2)\left(\frac{\alpha}{\beta^\alpha}\theta^\alpha - \alpha + 1\right)\right] \quad (22)$$

Similarly differentiating $D$ w.r.t the parameter $a$ results in:

$$\frac{\partial D}{\partial a} = E\left[2\theta x + x\theta^{-1}(1-\theta^2)\left(\frac{\alpha}{\beta^\alpha}\theta^\alpha - \alpha + 1\right) - \frac{1}{a}\right] \quad (23)$$

From the above equations we get the following on-line learning rule with stochastic gradient descent with learning rate $\eta$

$$\Delta b = -\eta\left[2\theta + \theta^{-1}(1-\theta^2)\left(\frac{\alpha}{\beta^\alpha}\theta^\alpha - \alpha + 1\right)\right]. \quad (24)$$

$$\Delta a = \frac{\eta}{a} + x\Delta b \quad (25)$$

*Note*: This relationship between the neuron parameter update rules ($\Delta a$ and $\Delta b$) is generic and valid irrespective of the neuron non-linearity or target probability distribution.

## References


Antonelo E, Schrauwen B, Stroobandt D (2008) Mobile robot control in the road sign problem using reservoir computing networks. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp 911–916

Bertschinger N, Natschläger T (2004) Real-time computation at the edge of chaos in recurrent neural networks. Neural Comput 16:1413–1436

Bernacchia A, Seo H, Lee D, Wang XJ (2011) A reservoir of time constants for memory traces in cortical neurons. Nat Neurosci 14(3):366–372

Boedecker J, Obst O, Mayer MN, Asada M (2009) Initialization and self-organized optimization of recurrent neural network connectivity. HFSP J 5:340–349

Buonomano DV, Laje R (2010) Population clocks: motor timing with neural dynamics. Trends Cogn Sci 14:520–527

Büsing L, Schrauwen B, Legenstein R (2010) Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. Neural Comput 22:1272–1311

Desai NS, Rutherford LC, Turrigiano GG (1999) Plasticity in the intrinsic excitability of cortical pyramidal neurons. Nat Neurosci 2:515–520

Ganguli S, Dongsung H, Sompolinsky H (2008) Memory traces in dynamical systems. Proc Natl Acad Sci USA 105:18970–18975

Jaeger H (2001) Short term memory in echo state networks. GMD Report 152, German National Research Center for Information Technology

Jaeger H (2003) Adaptive nonlinear system identification with echo state networks. In: Advances in Neural Information Processing Systems, pp 593–600

Jaeger H, Haas H (2004) Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science 2:78–80

Jaeger H, Lukosevicius M, Popovici D, Siewert U (2007) Optimization and applications of echo state networks with leaky-integrator neurons. Neural Netw 20:335–352

Jaeger H (2007) Discovering multiscale dynamical features with hierarchical echo state networks (Tech. Rep. No. 10). Jacobs University, Bremen

Li C (2011) A model of neuronal intrinsic plasticity. IEEE Trans Auton Ment Dev 3:277–284

Lizier TJ, Pritam M, Prokopenko M (2011) Information dynamics in small-world boolean networks. Artif Life 17:293–314

Lizier JT (2012) JIDT: an information-theoretic toolkit for studying the dynamics of complex systems. http://code.google.com/p/information-dynamics-toolkit/

Lizier TJ, Prokopenko M, Zomaya AY (2012) Local measures of information storage in complex distributed computation. Inf Sci 208:39–54

Lukosevicius M, Jaeger H (2009) Reservoir computing approaches to recurrent neural network training. Comput Sci Rev 3:127–149

Maass W, Natschläger T, Markram H (2004) Computational models for generic cortical microcircuits. In: Computational neuroscience: a comprehensive approach, chapter 18, pp 575–605

Manoonpong P, Kolodziejski C, Wörgötter F, Morimoto J (2013a) Combining correlation-based and reward-based learning in neural control for policy improvement. Adv Complex Syst (in press)

Manoonpong P, Parlitz U, Wörgötter F (2013b) Neural control and adaptive neural forward models for insect-like, energy-efficient, and adaptable locomotion of walking machines. Front Neural Circuits 7:12. doi:10.3389/fncir.2013.00012

Ozturk MC, Xu D, Prncipe JC (2007) Analysis and design of echo state networks. Neural Comput 19:111–138

Paleologu C, Benesty J, Ciochino S (2008) A robust variable forgetting factor recursive least-squares algorithm for system identification. IEEE Signal Process Lett 15:597–600

Ren G, Chen W, Kolodziejski C, Wörgötter F, Dasgupta S, Manoonpong P (2012) Multiple chaotic central pattern generators for locomotion generation and leg damage compensation in a hexapod robot. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 2756–2761

Schrauwen B, Wardermann M, Verstraeten D, Steil JJ, Stroobandt D (2008) Improving reservoirs using intrinsic plasticity. Neurocomputing 71:1159–1171

Shi Z, Han M (2007) Support vector echo-state machine for chaotic time-series prediction. IEEE Trans Neural Netw 18:359–372

Sompolinsky H, Crisanti A, Sommers HJ (1988) Chaos in random neural networks. Phys Rev Lett 61:259–262

Steingrube S, Timme M, Wörgötter F, Manoonpong P (2010) Self-organized adaptation of a simple neural circuit enables complex robot behaviour. Nat Phys 6:224–230

Sussillo D, Abbott LF (2009) Generating coherent patterns of activity from chaotic neural networks. Neuron 4:544–557

Tetzlaff C, Kolodziejski C, Markelic I, Wörgötter F (2012) Time scales of memory, learning, and plasticity. Biol Cybern 6:715–26

Triesch J (2007) Synergies between intrinsic and synaptic plasticity mechanisms. Neural Comput 4:885–909

Turrigiano G, Abbott LF, Marder E (1994) Activity-dependent changes in the intrinsic properties of cultured neurons. Science 264:974–977

Ungerleider LG, Courtney SM, Haxby JV (1998) A neural system for human visual working memory. Proc Natl Acad Sci USA 95:883–890

Yamashita Y, Tani J (2008) Emergence of Functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. PLoS Comput Biol 4(11):e1000220. doi:10.1371/journal.pcbi.1000220