# SYDE 556/750
# Simulating Neurobiological Systems
# Lecture 6: Recurrent Dynamics

Andreas Stöckel

Based on lecture notes by
Chris Eliasmith and Terrence C. Stewart

February 4 & 6, 2020

**Accompanying Readings: Chapter 8 of Neural Engineering**

# Contents

# 1 Introduction

**Note:** We have now discussed the first two principles of the Neural Engineering Framework: Representation and Transformation. These two principles allow us to build feed-forward networks, i.e., networks that can – mathematically speaking – be described as acyclic directed graphs.

Biological neural networks are dynamical systems – that is, they perform computation over time. The state of the network at a previous time influences the outcome of a computation.

So far, we have discussed two sources of dynamics: neuron models and synaptic filters. However, these dynamics usually possess relatively short time constants, that is, the system "forgets" its previous state quite quickly. For example, LIF neuron are reset to their initial state whenever they emit a spike, and synaptic time constants are usually in a range between 1 ms to 100 ms. In turn, that means that each population in a feed-forward will eventually "forget" events that happened more than a fraction of a second ago.

This is of course not what we observe in biological systems. Most animals appear to have memories of events in the past spanning seconds to years. In theory, there are three ways to implement such "memory":

- **More complex neuron models.** Biological neurons, and correspondingly more complex neuron models, typically posses more complex dynamics, such as *firing rate adaptation*. However, the effects of firing rate adaptation typically only last for a few seconds and are thus not sufficient to explain "memory" in biological systems.
- **Changing connection weights over time.** Another form of "memory" can be implemented by changing synaptic weights over time. In the context of the NEF we refer to this process as "learning".
- **Recurrent connections.** The feed-forward nature of the networks we've built so far is the primary reason why they quickly "forget" about their previous state. However, we could introduce "backwards" (or "recurrent") connections that "remind" the network their previous states.

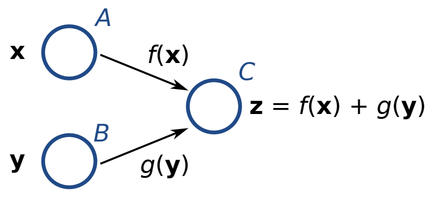
Ultimately, both learning and recurrent connections are valid solutions to the problem of our networks "forgetting" the history of their input. Typically, synaptic weight changes are regarded as evoking longer term changes (minutes to years), whereas recurrent connections are a model of shorter term changes in dynamics (seconds to minutes). In this lecture we focus on recurrence. We will discuss *learning* at a later point in time. In the Neural Engineering Framework, recurrent connections are used to realize the third principle:

**NEF Principle 3 – Dynamics**
Neural dynamics are characterized by considering neural representations as control theoretic state variables. We can use control theory (and dynamical systems theory) to analyse and construct these systems.

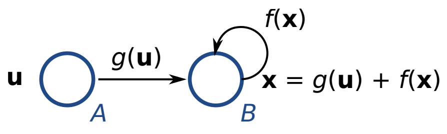
# 2 Exploring Recurrent Connections

Before we take a deeper dive into the mathematics of neural dynamics, we first discuss how we can implement recurrent connections in the context of an NEF network, followed by some experiments in which we explore what happens for certain functions that are being computed in the feedback function.

## 2.1 Implementing Recurrent Connections



**(a)** Feed forward network



**(b)** Recurrent network

**Figure 1:** Comparing a feed-forward and a recurrent neural network.

**Note:** When we talk about recurrent connections in the context of the Neural Engineering Framework, we typically refer to a neuron population that is connected back to itself. While this might seem a little strange, we are trying to point out that recurrent connections change virtually nothing in terms of the math that we have used so far.

**Revisiting a neural population receiving input from two pre-populations** As we discussed in the last lecture, fig. 1a depicts an ensemble $C$ receiving input from to pre-ensembles $A$ and $B$. Mathematically, each neuron $i$ in the post-population $C$ receives an input current $J_i^C(t)$

$$J_i^C(t) = \left\langle \mathbf{w}_i^A, (\mathbf{a}_{\mathrm{pre}}^A * h)(t) \right\rangle + \left\langle \mathbf{w}_i^B, (\mathbf{a}_{\mathrm{pre}}^B * h)(t) \right\rangle + J_i^{\mathrm{bias}}, \quad \text{where } \mathbf{w}_i^A = \left(\mathbf{E}\mathbf{D}^{g_1}\right)_i \text{ and } \mathbf{w}_i^B = \left(\mathbf{E}\mathbf{D}^{g_2}\right)_i,$$

where $\mathbf{a}_{\mathrm{pre}}^A$ and $\mathbf{a}_{\mathrm{pre}}^B$ are the activities of the pre-populations $A$, $B$, respectively. The function $h$ is the synaptic filter, $\mathbf{E}$ is a matrix of post-population encoders, $\mathbf{D}^{g_1}$ is the decoding the function $g_1(\mathbf{x})$ from population $A$, $\mathbf{D}^{g_2}$ is decoding the function $g_2(\mathbf{y})$ from the pre-population population $B$. Then, the value $\mathbf{z}$ represented by the population is approximately $\mathbf{z} = f(\mathbf{x}) + f(\mathbf{y})$.

**Recurrent populations** We can use the same setup as above to implement the simple recurrent connection depicted in figure fig. 1b. Here, a population $B$ represents a value $\mathbf{x}$ and receives input from both a pre-population $A$ representing a value $\mathbf{u}$ and itself. That is, each neuron $i$ in $B$ receives the current

$$J_i^B(t) = \left\langle \mathbf{w}_i^A, (\mathbf{a}_{\mathrm{pre}}^A * h)(t) \right\rangle + \left\langle \mathbf{w}_i^B, (\mathbf{a}_{\mathrm{pre}}^B * h)(t) \right\rangle + J_i^{\mathrm{bias}}, \quad \text{where } \mathbf{w}_i^A = \left(\mathbf{E}\mathbf{D}^{g}\right)_i \text{ and } \mathbf{w}_i^B = \left(\mathbf{E}\mathbf{D}^{f}\right)_i.$$

Notice that apart from names, there is virtually no difference between the previous two equations. The population $B$ approximately represents the value $\mathbf{x} = g(\mathbf{u}) + f(\mathbf{x})$. In terms of control theory, the variable $\mathbf{u}$ is the *input* to our system, whereas the variable $\mathbf{x}$ is the represented value. The function $g(\mathbf{x})$ is a function transforming our input, the function $f(\mathbf{x})$ is the *feedback function*.

**Note:** When implementing a computer simulation that contains recurrent connections, each occurrence of a population pre-activity $\boldsymbol{\alpha}_{\mathrm{pre}}$ should be interpreted as the pre-population activity from the last timestep.

Note that this will introduce a delay of one timestep each time a new set of populations is connected. While not much of an issue in general, note that this delay is not part of the continuous equations that are being simulated. The implementation of the Neural Engineering Framework in *Nengo* is thus a little more clever and will only introduce delays if it has to; it will not introduce a delay on pure feed-forward connections.

## 2.2 Experimenting with Recurrent Connections

The above equations illustrate how recurrent connections are implemented and what the represented value of the recurrently connection is. To get some intuition for the dynamics of such a system, we can try test different feedback functions $f(\mathbf{x})$. Figure 2 depicts the behaviour of a population of LIF neurons representing a one-dimensional value along with various feedback functions and various synaptic filter time constants.

**Note:** The synaptic filter $h(t)$ we are using here is the exponential low pass

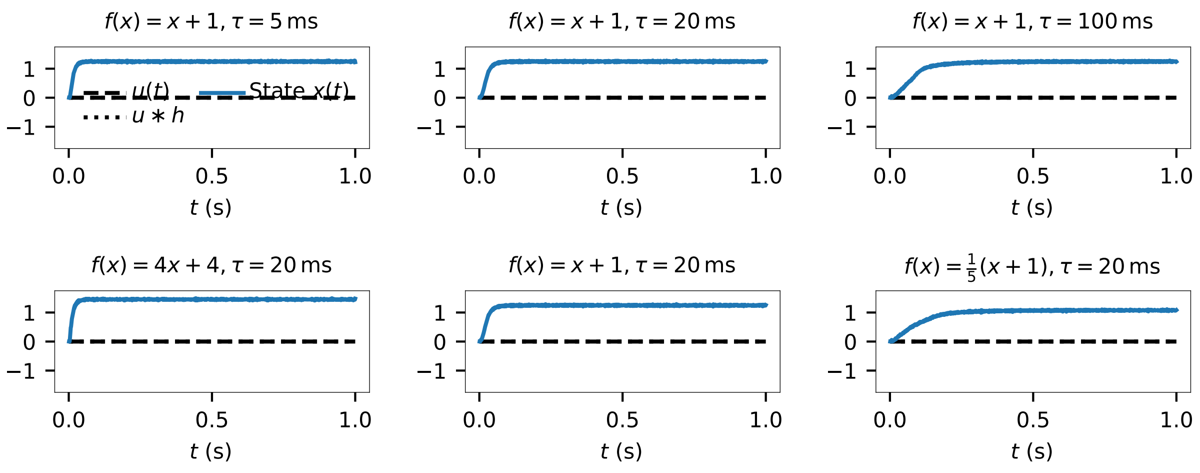
$$h(t) = \begin{cases} \frac{1}{\tau} e^{-t/\tau} & \text{if } t \geq 0\,, \\ 0 & \text{if } t < 0\,. \end{cases} \tag{1}$$

This a version of the general synaptic filter discussed in lecture four (in particular $n = 0$). For this specific case, we can compute the normalisation factor $c$ in closed form. It holds $c = \frac{1}{\tau}$. To see that this factor is correct, we can compute the integral

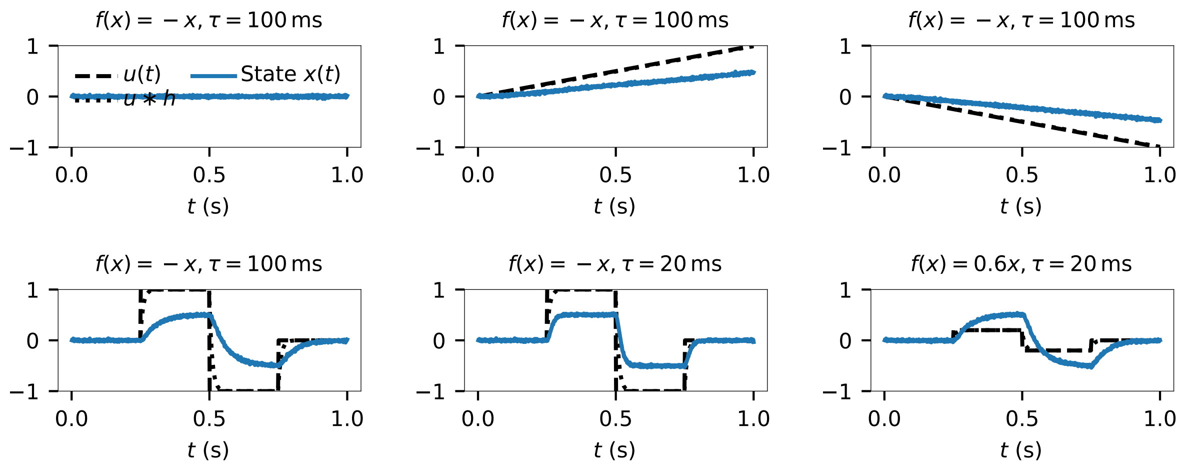
$$\int_0^\infty h(t)\,\mathrm{d}t = \int_0^\infty \frac{1}{\tau} e^{-t/\tau}\,\mathrm{d}t = \frac{1}{\tau}\big[-\tau e^{-t/\tau}\big]_0^\infty = \frac{1}{\tau}\big[0 + \tau\big] = 1\,.$$

**Feedback $f(x) = x + 1$** The dynamics of the system for a feedback function $f(x) = x + 1$ are depicted in fig. 2a. The resulting system converges to a value slightly above one. The rate of convergence is determined by the time constant $\tau$ and the scaling of the feedback function; an increase of the time constant by a factor $c$ seems to result in similar dynamics as dividing the feedback function $f$ by $c$.

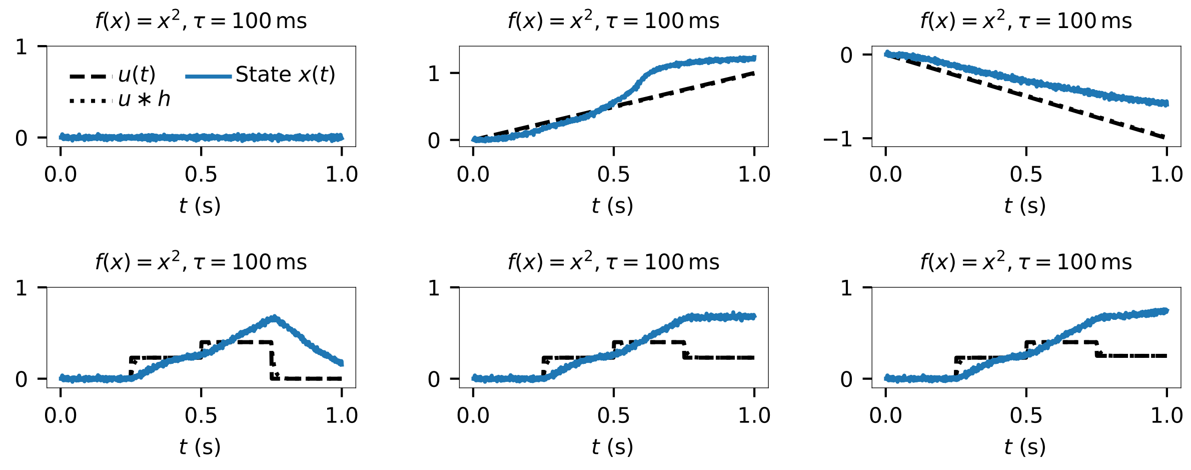
**Note:** This system only converges to a fixed value due to saturation of the neural responses. For example, if we replace the LIF neurons with rectified linear units, the state $x$ will diverge monotonically towards infinity.



**(a)** Exploring the feedback function $f(\mathbf{x}) = x + 1$



**(b)** Exploring the feedback function $f(\mathbf{x}) = -x$



**(c)** Exploring the feedback function $f(\mathbf{x}) = x^2 s$

**Figure 2:** Exploring the effect of different feedback functions $f(\mathbf{x})$, input signals $u(t)$, and synaptic time constants $\tau$ on the dynamics of a single-population network. 100 LIF neurons, decoded data are filtered with a 10 ms time-constant exponential low-pass

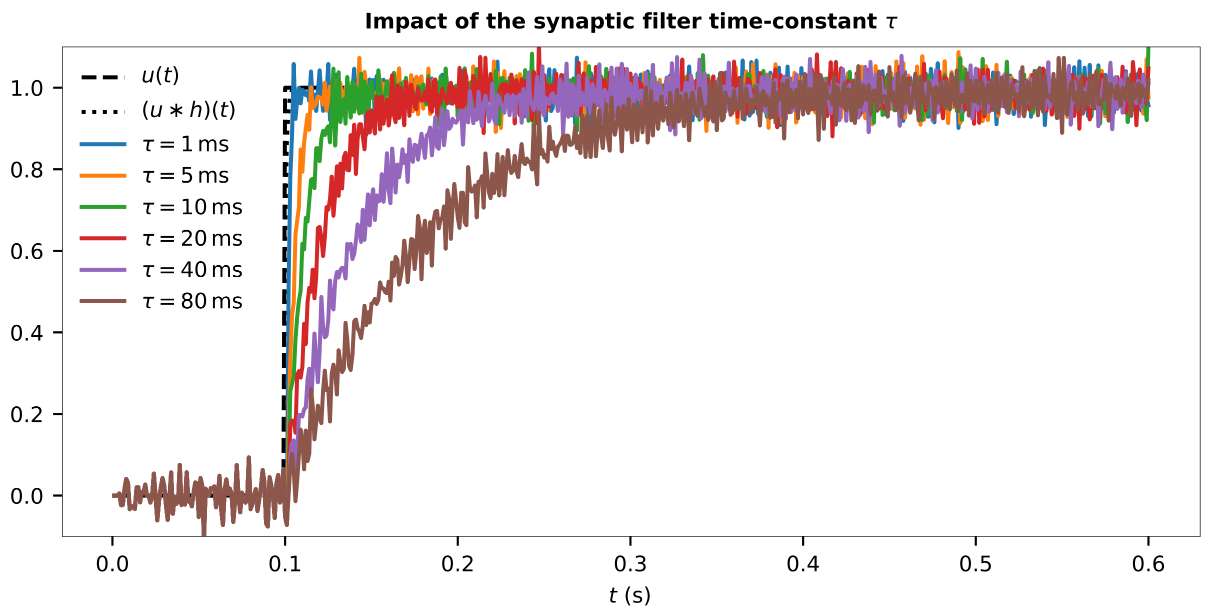


**Figure 3:** Dynamics of a feed-forward system for varying synaptic time constants $\tau$. 1000 LIF neurons, decoded data are filtered with a 1 ms time-constant exponential low-pass.

**Feedback $f(x) = -x$** The dynamics of a system with the feedback function $f(x) = -x$ are depicted in fig. 2b. The system converges to exactly $\frac{1}{2}$ of the input $u(t)$. Rearranging the function describing the represented value $x$ yields

$$x = g(u) + f(x) = u - x \Leftrightarrow x = \frac{u}{2}.$$

This explains why the function converges to a steady state of $\frac{u}{2}$; however, this analysis does not help us to understand the dynamics of the system (bottom half of fig. 2b).

**Feedback $f(x) = x^2$** The dynamics for a feedback function $f(x) = x^2$ are depicted in fig. 2c. The system converges to $u(t)$ for inputs $u(t) \lessapprox 0.4$ but diverges for larger values. Interestingly, once the system is in a state with $x > 0.4$, it will not converge back to $u(t)$ for values of $u(t)$ between approximately 0.2 and 0.4. For negative inputs $u(t)$ the system approximately converges to $\frac{1}{2}u(t)$.

## Mathematical Analysis

**Note:** While the above examples give us a feeling for what the dynamics of a neural system with recurrent connection might look like, these systems remain quite mysterious. We need to perform some mathematical analysis in order to truly understand these networks.

In order to gain a better understanding of recurrent dynamics, let's first revisit a simple feed-forward network. Figure 3 shows a single neuron population with varying synaptic filter time-constants $\tau$. As we can see, the dynamics of the system almost exclusively depend on the filter time constant. Furthermore, as we have already discussed, it does not matter whether we apply the filter before or after the encoding process. Correspondingly, it is sufficient to analyse dynamics in terms of represented values and to ignore individual neurons.

**Note:** *Assumptions.* As so often, we will ignore the fact that we are dealing with vectorial quantities in the following subsection and instead assume that $d = 1$. That being said, all equations are also holding for higher-dimensional representations.

Furthermore, we will assume that the synaptic filter is the same for both the synapses from the pre-populations, as well as the synapses filtering the recurrent connections. Often, we will find that the recurrent connections have a longer time constant than the input connections. However, the math we derive will still work reasonably well.

## Analysing Neural Network Dynamics

The equation detailing the recurrence relationship $x = g(u) + f(x)$ did not include the synaptic filter. Assuming that both the input and the recurrent connections are filtered by the same filter we get the following dynamical system:

$$x(t) = \big(h \ast \big(g(u) + f(x)\big)\big)(t) = \int_{-\infty}^{\infty} h(t')\big(g(u(t-t')) + f(x(t-t'))\big)\,\mathrm{d}t' \,.$$

Unfortunately, and as we have seen in previous lectures, the convolution operator is a little unwieldy. We could eliminate the convolution by switching to the Fourier Domain, however, there is an alternative, more powerful alternative: the Laplace transformation $\mathcal{L}\{f\} = F(s)$.

**Note:** The Laplace transformation is a more general version of the Fourier transformation.

$$\mathcal{L}\{f\} = F(s) = \int_{0}^{\infty} f(t) e^{-st}\,\mathrm{d}t \,,$$

where $s$ is a complex-valued parameter $s = \sigma + i\omega$. Notice that the Laplace and Fourier transformation are almost the same if we let $\sigma = 0$; the major remaining difference are the integration boundaries. Since the lower integration boundary of the Laplace transform is zero (instead of $-\infty$ in the case of the Fourier transformation), the Laplace transformation will ensure causality.

The Laplace and Fourier transformation share many commonalities, including linearity, and the fact that convolution becomes multiplication. However, it also holds

$$\mathcal{L}\left\{\frac{\mathrm{d}f}{\mathrm{d}t}\right\} = \mathcal{L}\{f'\} = sF(s) \,,$$

that is, a time-differental just turns into multiplication with the variable $s$. This makes the Laplace transformation an important tool for dealing with differential equations – differentiation is just turned into multiplication with a variable.

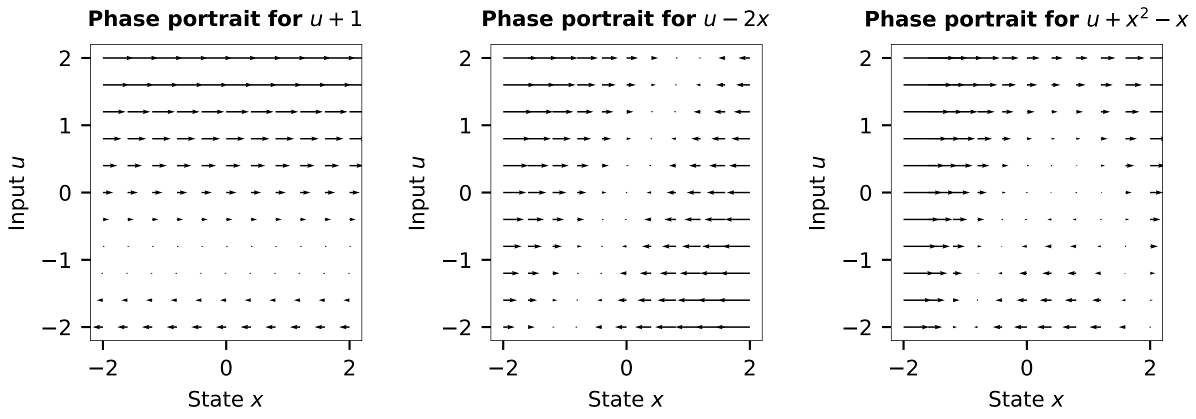


**Figure 4:** Phase portraits(see note) for the dynamical systems corresponding to the three feedback functions we experimented with.

The Laplace-transform of the first-order exponential low-pass filter in eq. (1) becomes:

$$\begin{aligned}\mathcal{L}\{h\} &= \int_0^\infty h(t)e^{-st}\,\mathrm{d}t = \int_0^\infty \frac{1}{\tau}e^{-t'/\tau}e^{-st'}\,\mathrm{d}t' = \int_0^\infty \frac{1}{\tau}e^{-t'(1/\tau+s)}\mathrm{d}t' \\ &= \left[-\frac{1}{\tau\left(\frac{1}{\tau}+s\right)}\right]_0^\infty = \frac{1}{1+s\tau} = H(s)\,.\end{aligned}$$

Hence, we can rewrite the dynamical system we were looking at in the Laplace domain

$$\begin{aligned} X(s) &= H(s)\big(G(s)+F(s)\big) \\ \Leftrightarrow X(s)(1+s\tau) &= G(s)+F(s) \\ \Leftrightarrow sX(s) &= \frac{1}{\tau}\big(G(s)+F(s)-X(s)\big)\,.\end{aligned}$$

Converting back to the time domain we get the following differential equation.

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \frac{1}{\tau}\big(g(u(t))+f(x(t))-x(t)\big)\,.$$

**Note:** *Canonical dynamical system and phase portrait.* The canonical form of time-independent a dynamical system

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \phi(u(t),x(t))$$

where $\phi(u,x)$ is an arbitrary function mapping an input $u$ and the state $x$ onto a state differential. We can draw the function $\phi(u,x)$ as a so called *phase portraits* by drawing an arrow corresponding to the state update in an $x$-$u$ coordinate system.

## 2.3 Analysis of the Previous Experiments

**Analysing $f(x) = x + 1$** For this particular feedback function, our dynamical system becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \frac{\mathrm{d}}{\mathrm{d}t}\phi(x(t), u(t)) = \frac{1}{\tau}\big(u(t) + x(t) + 1 - x(t)\big) = \frac{1}{\tau}\big(u(t) + 1\big).$$

Looking at the phase portrait for $f(u, x) = u - 2x$ in fig. 4, we can see that this function will diverge for inputs $u(t) \neq -1$.

**Analysing $f(x) = -x$** For this feedback function, our dynamical system becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \frac{\mathrm{d}}{\mathrm{d}t}\phi(x(t), u(t)) = \frac{1}{\tau}\big(u(t) - 2x(t)\big),$$

which can be re-written as

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \frac{\mathrm{d}}{\mathrm{d}t}\phi(x(t), u(t)) = \frac{2}{\tau}\left(\frac{u(t)}{2} - x(t)\right),$$

Looking at the phase portrait for $f(u, x) = u - 2x$ in fig. 4, we can see that this system has an line-attractor at $x = \frac{u}{2}$.

**Analysing $f(x) = x^2$** For this feedback function, our dynamical system becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \frac{\mathrm{d}}{\mathrm{d}t}\phi(x(t), u(t)) = \frac{1}{\tau}\big(u(t) + x(t)^2 - x(t)\big).$$

This system can be re-written as a dynamical system with two attractors

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \frac{1}{\tau}\left(x(t) - 1 + \frac{\sqrt{1 - 4u(t)}}{2}\right)\left(x(t) - 1 - \frac{\sqrt{1 - 4u(t)}}{2}\right).$$

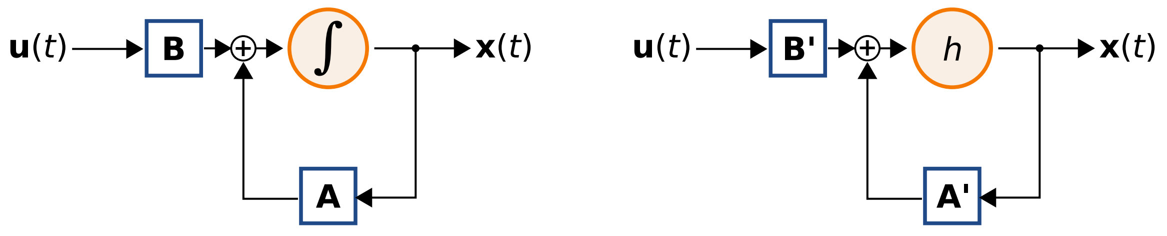
These two attractors are visible in the phase portrait for $\phi(u, x) = u + x^2 - x$ (fig. 4); however, we can see that one attractor (on the left side of the portrait) is stable, whereas the other attractor is unstable (right side).

## 2.4 Implementing Arbitrary Time-Invariant Dynamical Systems

We have now seen how we can analyse the dynamics emerging from a certain input and feedback function pair $g(\mathbf{u})$, $f(\mathbf{x})$. In other words, given a feedback function $f(\mathbf{x})$ we can find the corresponding dynamical system $\phi(\mathbf{u}, \mathbf{x})$. This raises the question whether we can also inverse this process, i.e., given a dynamical system $\phi(\mathbf{u}, \mathbf{x})$ find the correct feedback and input function $f(\mathbf{x})$, $g(\mathbf{u})$ that implement this dynamical system in a neural context.

**Transforming a linear time-invariant system** We will first look at a special case, a linear time-invariant dynamical system (LTI). This is a dynamical system of the form

$$\phi(\mathbf{u}, \mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u},$$



**(a)** Canonical LTI system **(b)** Neural LTI system

**Figure 5:** Comparison between an LTI system being evaluated using an integrator, compared to the corresponding neural implementation. Our goal is to find matrices $\mathbf{A}'$ and $\mathbf{B}'$ such that the two systems are equivalent.

where **A** is a matrix describing the feedback and **B** is a matrix describing a mapping from the input onto the state differential.

Normally, we would implement such a system using a perfect integrator, that is we compute the sum $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ giving us a state differential. Integration of the differential gives us a new state, which is fed back for the computation of the next state differential (fig. 5a).

However, when building neural networks, we do not have access to a perfect integrator. Instead, we have the synaptic filter $h$, which can be seen as a "leaky integrator". Rephrasing the general question we were asking above, we are looking for a way to rewrite the dynamical system $\phi(\mathbf{u}, \mathbf{x})$ into a new dynamical system $\phi'(\mathbf{u}, \mathbf{x})$, such that $\phi'$ is equivalent to $\phi$ in the context of a leaky integrator (fig. 5b). In particular, we would like to find another LTI system

$$\phi'(\mathbf{u}, \mathbf{x}) = \mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{u}\,.$$

Writing the canonical dynamical system and the neural dynamical system in the time domain we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)\,, \qquad \mathbf{x}(t) = \big(h * \big(\mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{u}\big)\big)(t)\,.$$

We would like to find $\mathbf{A}'$ and $\mathbf{B}'$ such that the two systems behave in the same way. Converting to the Laplace to eliminate the convolution we get

$$sX(s) = \mathbf{A}X(s) + \mathbf{B}U(s)\,, \qquad X(s) = H(s)(\mathbf{A}'X(s) + \mathbf{B}'U(s))\,.$$

Let's expand the right equation and bring it into the same form as the left equation:

$$\begin{aligned}
X(s) &= \frac{1}{1+s\tau}(\mathbf{A}'X(s) + \mathbf{B}'U(s)) \\
\Leftrightarrow X(s)(s\tau + 1) &= \mathbf{A}'X(s) + \mathbf{B}'U(s) \\
\Leftrightarrow s\tau X(s) &= \mathbf{A}'X(s) + \mathbf{B}'U(s) - X(s) \\
&= (\mathbf{A}' - \mathbf{I})X(s) + \mathbf{B}' \\
\Leftrightarrow sX(s) &= \frac{1}{\tau}(\mathbf{A}' - \mathbf{I})X(s) + \frac{1}{\tau}\mathbf{B}'\,.
\end{aligned}$$

Comparing to our target equation we get

$$\mathbf{A} = \frac{1}{\tau}\big(\mathbf{A}' - \mathbf{I}\big)\,, \qquad \mathbf{B} = \frac{1}{\tau}B'\,.$$

Rearranging yields

*(Neural Engineering Framework LTI)*

$$\mathbf{A}' = \tau\mathbf{A} + \mathbf{I}\,, \qquad \mathbf{B}' = \tau\mathbf{B}\,. \tag{2}$$

**Transforming an additive time-invariant dynamical system** We can apply the same math to a time-invariant dynamical system $\phi$ that can be decomposed into an addition between $g(\mathbf{u})$ and $f(\mathbf{x})$, that is $\phi(\mathbf{u},\mathbf{x}) = g(\mathbf{u}) + f(\mathbf{x})$, with the target system $\phi'(\mathbf{u},\mathbf{x}) = g'(\mathbf{u}) + f'(\mathbf{x})$.

$$\begin{aligned} X(s) &= H(s)\big(G(U(s)) + F(X(s))\big) \\ \Leftrightarrow sX(s) &= \frac{1}{\tau}\big(F(X(s)) - X(s)\big) + \frac{1}{\tau}G(X(s))\,. \end{aligned}$$

Transforming back to the time domain, comparing to the target system and rearranging yields

$$f'(\mathbf{x}) = \tau f(\mathbf{x}) + \mathbf{x}\,, \qquad g'(\mathbf{u}) = \tau\mathbf{u}\,.$$

🚧 UNDER CONSTRUCTION 🚧

## 2.5 Examples

🚧 UNDER CONSTRUCTION 🚧

### Integrator

$$\begin{aligned} \mathbf{A} &= 0 & \mathbf{B} &= \mathbf{I} \\ \Leftrightarrow \mathbf{A}' &= \mathbf{I} & \Leftrightarrow \mathbf{B}' &= \tau\mathbf{I} \end{aligned}$$

### Oscillator

$$\begin{aligned} \mathbf{A} &= \begin{pmatrix} 0 & -\omega \\ \omega & 0 \end{pmatrix} & \mathbf{B} &= 0 \\ \Leftrightarrow \mathbf{A}' &= \begin{pmatrix} 1 & -\tau\omega \\ \tau\omega & 1 \end{pmatrix} & \Leftrightarrow \mathbf{B}' &= 0 \end{aligned}$$

### Gated Integrator

$$\begin{aligned} f(\mathbf{x}) &= 0 & g(\mathbf{u}) &= \alpha\mathbf{u} \\ f'(\mathbf{x}) &= \mathbf{I} & \Leftrightarrow g(\mathbf{u}) &= \tau\alpha\mathbf{u} \end{aligned}$$

**Controlled Oscillator**

$$\mathbf{A} = \begin{pmatrix} 0 & -\alpha\omega \\ \alpha\omega & 0 \end{pmatrix} \qquad \mathbf{B} = 0$$

$$\Leftrightarrow \mathbf{A}' = \begin{pmatrix} 1 & -\alpha\tau\omega \\ \alpha\tau\omega & 1 \end{pmatrix} \qquad \Leftrightarrow \mathbf{B}' = 0$$

## 3 Examples of Dynamics in Biological Systems

🚧 UNDER CONSTRUCTION 🚧