Coupled Local Supervised Learning Notes

Monica Conte

January 2023

In this document I briefly review the theory on *Coupled Local Supervised Learning* (CLSL) as explained in [1] and applied in [2]. This method constitutes a novel approach in the design of materials that possess the ability of learning.

The goal of this work is to physically realize a system that autonomously learns to accomplish tasks (such as allostery) in an efficient and accurate way. A similar system has already been implemented in [2] by building a network of variable resistors. In Section 4 we review this approach and underline its limitations.

In order to build a better system, we decide to substitute the variable resistors with memristors, with the intuition that their properties of retaining memory and changing continuously would improve the training. We choose to work with conical micorfluidic channels [3], as they are easy to fabricate, cheap and versatile.

Contents

1 Learning in Physical Systems

As opposed to materials designed with particular sets of functions, CLSL approach allows to build systems that don't possess explicit information about the desired functionality but physically adapts to applied external forces and develop the ability to perform tasks. This method belongs to a class of strategies based on learning, where systems can adjust or be adjusted microscopically in response to training set to develop desired functionalities. Until recently, such idea was primarily applied in the context of non-physical networks, like artificial neural networks (ANN).

There are two types of learning:

- **Global Learning** involves the minimization of a global function and the subsequent *tuning* of the leaning degrees of freedom. **An external intervention at microscopic level is required.**
- In **Local Lerning** the evolution is **autonomous and requires no external designer** for the evaluation of the current state and subsequent modification. This is particularly useful in physical systems, since their microscopical elements do not perform computations and do not encode a priori information on the desired functionality.

CLSL implements local learning on physical networks with the aim of training it to achieve desired responses on *target nodes* as a response to external constraints applied to *source nodes*. The method is inspired by *contrastive learning* and by the strategy of *equilibrium propagation*.

1.1 ANN framework

The building blocks of ANN are *neurons*. They are elements that take inputs and produce outputs by performing some mathematical operations on the inputs. Fig. 1 shows a neuron with two inputs x_1 and x_2. The output depends on the inputs, the weight and the bias. Training a neuron means modifying the weights in such a way that it outputs a desired value.

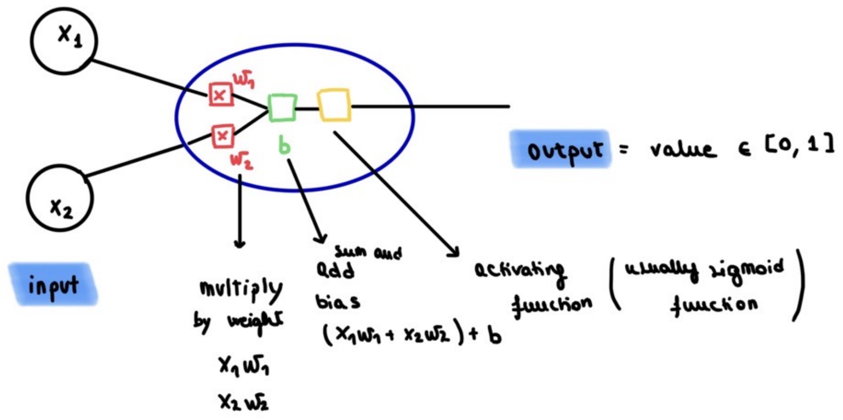

Figure 1: Two inputs (x_1 and x_2) neuron. Blue circle contains the operations performed in a neuron: **multiply inputs to weights** associated to the links, **sum inputs and add a bias** b, **apply an activating function**. The outcome is a value belonging to $[0, 1]$.

Neural networks are structures made of many neurons. These neurons are organized in layers, an example in Fig. 2. There are three types of layers: input layer, hidden layers and output layers. Input layers read the inputs, operations are performed in the hidden layer and output is generated in output layer. Hence, a neural network can be seen as a set of *nodes* (the neurons) connected by *edges* (the links with weights). Given an

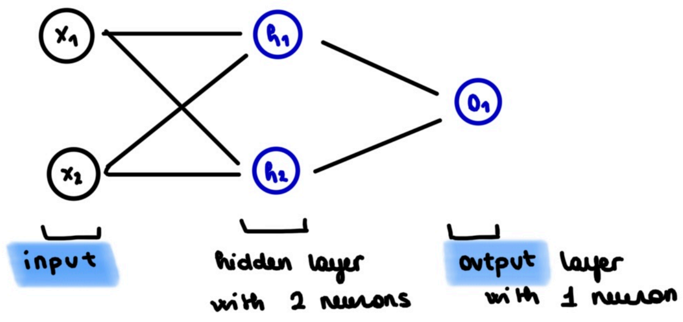

Figure 2: Example of a neural network: a structure made of neurons organized in layers.

input, each neuron in each layer performs computations and gives output that are inputs for the following neuron. This process of proceeding from the inputs to the outputs is called **feedforward**.

A neural network needs *training* in order to be able to perform a task. The standard way to train a network is by using a *loss function*, which quantifies how good the output of the network is compared to the desired output. Often, the mean-squared error (MSE) is used

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_{\text{desired}} - y_{\text{predicted}})^2$$

where the sum is take over the training samples and y indicates the value of the output in a specific neuron. From this simple formula we can already understand that a **smaller error implies a better prediction**.

The training method proceeds in changing the weights in the network in the direction of the minimization of the loss function. It is therefore useful to denote as $L(\mathbf{w})$ the loss function as a function of the set of weights $\mathbf{w}$. In order to understand how the loss function changes when the weight are modified, we consider the derivative of the loss function with respect to the weights. Let us take for example the first weight w_1, the derivative $\partial L/\partial w_1$ is found by applying the chain rule

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{\text{predicted}}}\frac{\partial y_{\text{predicted}}}{\partial w_1}$$

which is applied again to the second therm on the RHS following a path in the neural network which traces back to the location of the weight w_1. This procedure is called **backpropagation** since it accounts in propagating back the error in the network and identify the biggest sources of error.

Once the derivative of the loss function is calculated, one needs to find a way of optimizing the network accordingly. A possible way of minimizing the loss function is by using **gradient descent** method. The weights are updated as follows:

$$w \quad \to w - \eta\frac{\partial L}{\partial w}$$

and the logic behind is that it changes the weight towards the direction of descent. In fact, a positive derivative means that a positive increase of the weight increases the loss function, from which a negative sign in front of the term. Vice versa for the negative sign.

The training proceeds in steps and in each step the weights are optimized to decrease the error. There are many training samples in a dataset and a common training procedure randomly selects, at each step, one of the samples in the set and performs gradient descent. This procedure is called **stochastic gradient descent**.

1.2 CLSL framework

We now consider a physical neural network (PNN). It is, again, a structure made of nodes and edges but, as opposed to ANN, they are not organized in layers and in principle there is no fixed pattern. However, just like an ANN, we divide the nodes in *source nodes*, *hidden nodes* and *target nodes*. A generic PNN is shown in Fig. 3. These elements assume values of physical quantities and can be realized in real life. For example, in electric networks, the nodes assume values of potential and the edges of resistances. The training goal here

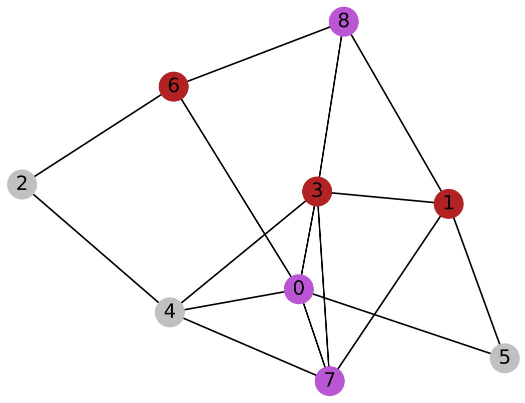

Figure 3: Example of a generic PNN. $5, 6, 7$ red dots denote input nodes, $0, 1, 3$ gray nodes denote hidden nodes and $2, 4, 8$ purple nodes denote target nodes.

is to train the network to give desired outputs in the output nodes given certain inputs at the source nodes, just like in the ANN.

In a PNN, the values on the nodes follow from the minimization of an *energy function*, as opposed to the feedforward procedure of ANN. For example, in electric networks, the function minimized is the power dissipated by the resistances. This, in turn, corresponds to satisfying Kichoff's law on current in the circuit of nodes and resistances, i.e. conservation of charge. In flow networks, minimization of power means conservation of flow.

1.2.1 General formulation of the method

Let us consdier a network with nodes indicized by μ and edges indicized by j. The *physical degrees of freedom* (pof) belong to the set of variables $\{x_\mu\}$. The *learning degrees of freedom* (lof) belong to the set w_j and play the roles of wieghts connecting the nodes. Given an initial condition $x_\mu(t=0)$ (part of the x_μ in the set given) and a set of weights, the physical degrees of freedom evolve to a steady state tha minimizes a *physical cost function* $E(x_\mu; w_j)$. Note that both $\{x_\mu\}$ and E depend on $\{w_j\}$.

Inspired by *contrastive Hebbian learning* (CHL), one considers two states of the network. The *free state* x_μ^F in which the source nodes x_S are constrained while the target x_T^F and hidden x_H^F equilibrate to the steady state. The *clamped state* x_μ^C in which the source x_S and the target x_T^C nodes are constrained while only the hidden nodes x_H^C equilibrate to the steady state. In the clampd state, the target nodes are nudged by an *external supervisor* towards the desired X_T values of outputs

$$x_T^C = x_T^F + \eta \left[X_T - x_T^F \right]$$

with $\eta \ll 1$. In contrast with CHL, the nudge parameter is small, which fixes the problem of free and clamped state reaching two different local minima of the energy function.

The authors propose the following updating rule for the lof:

$$\dot{\mathbf{w}} = \alpha \eta^{-1} \partial_{\mathbf{w}} \left\{ E(x_S, x_H^F, x_T^F) - E(x_S, x_H^C, x_T^C) \right\}$$

- α is a scalar **learning rate**
- **η is the nudging aplitude.**

The motivation behind this rule comes from the fact that we consdier a **physical** network, where generally one can partition the total energy E as a sum over edegs $E = \sum_j E_j(x_\mu(j); w_j)$ of energies E_j that only depend on the pof on the nodes that connect the edge j. Therefore, the learning rule is **local**. But does this rule effectively trains the network? The autors show (Appendix A in [1]) that this updating scheme minimizes an *effective cost function* that is not identical to the standard cost function but shares important features with it. They argue that minimizing the effective function mimics the minimization of a standard cost function. From how well the networks are trained, they observe that this affirmation is plausible. The CLSL approach shares many similarities with Equilibrium Propagation(EP) and the learning rule has the same form. However, in EP the learning rule minimizes an actual cost function.

Summarizing, the learning approach involves the minimization of an energy function and the implementation of the proposed updating rule. The former is motivated by physical plausability and the latter by inspiration from EP.

2 Application to Electrical Netwroks

The framework above can be applied in the context of electrical networks [2], in which nodes are connected by *variable resistors*. When input voltages are applied at the source nodes V_S, the voltages at the output V_O and hidden V_H nodes are determined such that the total power dissipated by the resistances $\mathcal{P}$ is minimized

$$\mathcal{P} = \frac{1}{2} \sum_j \frac{\Delta V_j^2}{R_j}$$

where j runs over the resistances R_j and ΔV_j is the potential difference between the nodes that the resistance links.

We define the two types of states, as above. In the free state the source nodes V_S^F are constrained, while V_H^F and V_T^F are determined by minimizing $\mathcal{P}$. In the clamped state, the source nodes V_C^F are fixed to the source values, the target nodes are clamped

$$V_T^C = V_C^F + \eta \left(V^D - V_C^F \right)$$

towards the desired output voltages V^D, while the hidden nodes are left to minimize $\mathcal{P}$. The parameter $0 < \eta \leq 1$ quanitifies the importance of the nudge.

The final goal is to deliver the desired output V^D to the target nodes. To do so, the network gets optimized by updating the values of the resistances following a certain *learning rule*. The adaptation of the learning rule of CLSL in the context of electrical circuits would lead to the following **updating rule for resistances**

$$\boxed{\dot{R}_j = \frac{\alpha}{2\eta R_j^2} \left[\left(\Delta V_j^C\right)^2 - \left(\Delta V_j^F\right)^2 \right]} \tag{1}$$

where R_j is the selected resistance to update, ΔV_j^C is the potential difference between the nodes that it connects in the clamped state and ΔV_j^F the difference in the free state. In the paper [2] they define

$$\gamma = \frac{\alpha}{2\eta}$$

as the learning rate.

In the experiment, they have discrete resistors and the rule above gets approximated with the following

$$\boxed{\dot{R}_j = \begin{cases} +\delta R & \text{if } |\Delta V_j^C| > |\Delta V_j^F| \\ -\delta R & \text{otherwise} \end{cases}} \tag{2}$$

which consists in taking the sign of Eq. (1) and multipy it by the value δR.

We can also reformulate the theory in terms of conductances $g_j = 1/R_j$, the updating rule of wihch can be derived from the power dissipated

$$\mathcal{P} = \frac{1}{2}\sum_j \frac{\Delta V_j^2}{R_j} = \frac{1}{2}\sum_j g_j \Delta V_j^2.$$

In terms of the learning rate γ, the **updating rule for conductances** reads

$$\boxed{\dot{g}_j = \gamma\left[\left(\Delta V_j^F\right)^2 - \left(\Delta V_j^C\right)^2\right]} \tag{3}$$

2.1 Simple Start: Voltage Divider

One of the simplest networks we can consider is a voltage divider, shown in Fig. 4. It has three nodes, two

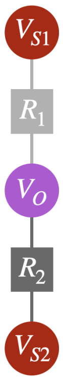

Figure 4: Schematic representation of the voltage divider considered in this section. It has two input nodes (red dots) with voltages V_{S1} and V_{S2}, one output node with voltage V_O and two edges with resistances R_1 and R_2.

of type source and one of type output with potential, respectively, V_{S1}, V_O and V_{S2}. These three nodes are connected by two resistances, R_1 and R_2, as in figure. The goal in training this network is to deliver a desired voltage of $V_D = 4V$ on the output node with voltage V_O, when imposing voltages of $V_{S1} = 5V$ and $V_{S2} = 0V$. At the beginning, the resistances are initiated to a value of $R = 50k\Omega$. In this first section, we are trying to reproduce the results shown in the paper [2], we are thus adopting the same setting and choice of parameters.

In the free state, only the source voltages are imposed while the value of the output voltage V_O minimizes the total power dissipated by the two resistances in the system,

$$\mathcal{P} = \frac{\Delta V_1^2}{R_1} + \frac{\Delta V_2^2}{R_2}$$

where $\Delta V_1 = V_{S1} - V_O$ and $\Delta V_2 = V_O - V_{S2}$. For this simple system, it is possible to explicitly solve $\partial\mathcal{P}/\partial V_O = 0$ finding the relation

$$R_2(V_1 - V_O) = R_1(V_O - V_2)$$

which suggests that the ratio between the resistances in the steady state is

$$\frac{R_1}{R_2} = \frac{V_1 - V_O}{V_O - V_2}. \tag{4}$$

In the case of the parameters we choose at the beginning of this section, we can state that when the network reaches the desired configuration, the ratio between the resistances should be $R_1/R_2 = 5/4 = 1.25$.

The training consists in steps, each of which results in an update of the edges in the network following the updating rule chosen. To reproduce the results in the paper [2], we first select the updating rule Eq. (2). This rule is discrete, it can update with a value of either $+\delta R$ or $-\delta R$, as opposed to a continuous rule, which updates with a value that in principle can be any real number. Before looking at the results of the training process on this system, we introduce the mean-squared error (MSE)

$$C = \sum_{i \in \text{output nodes}} (V_i - V_{D,i})^2,$$

this is a parameter that quantifies how far the output voltages V_i are from their desired values. Thus, the goal in the training is to reduce the MSE as much as possible.

We show the results of the training in Fig. 5 and we want to compare them with the first 100 training steps shown in Fig. 2(c) of the paper [**?**]. We observe that the training reduces the MSE of four orders of

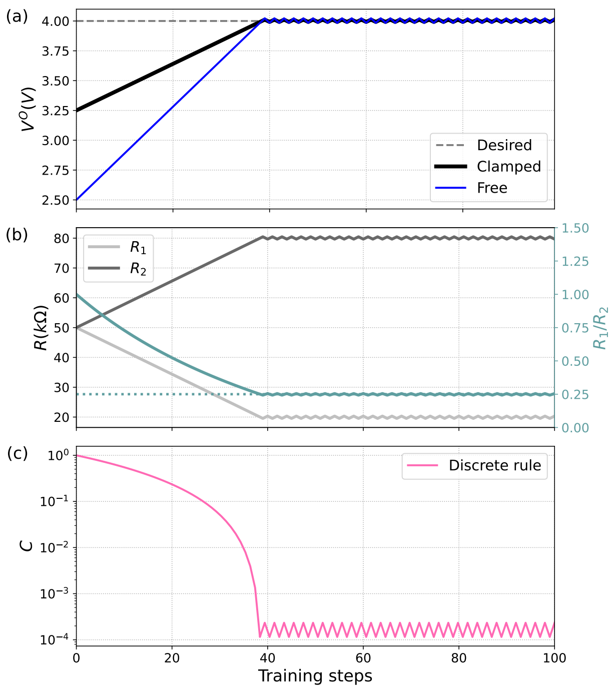

Figure 5: Training of the voltage divider in Fig. 4 with $V_{S1} = 5V$, $V_{S2} = 0V$, $V_D = 4V$ and discrete rule of Eq. (2). The clamping rate is set to $\eta = 0.5$. In (a) the plot of the voltage measured in the output node, V_O, during training, in the free state, clamped state. The dashed line indicates the desired voltage V_D. (b) Shows the values of the resistances in the edges R_1 and R_2 during training. The green line represents the ratio R_1/R_2, while the dashed green line the desired ration that follows from Eq. (4). In (c) the plot of the MSE.

magnitude in 40 steps, with a behaviour fairly similar to the results in the paper. It is after this value that we can notice the limitation of this discrete rule: the system is not adaptive. In fact, as the network gets trained and difference between the potentials in the free and clamped state

$$|\Delta V_j^C| - |\Delta V_j^F| = \sigma$$

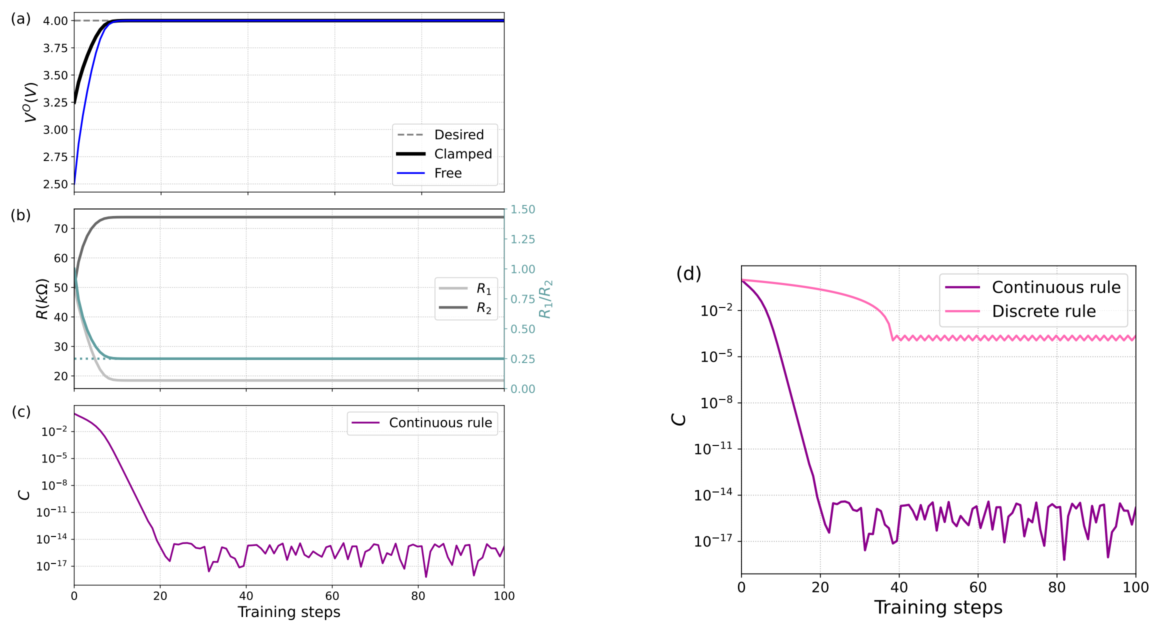

Figure 6: (a), (b), (c) Training the voltage divider with continuous rule on resistances. The nudging parameter $\eta = 0.5$ and $\alpha = 5 \times 10^3$. (d) Plot of the MSE resulting from implementing the discrete rule in Eq. (2) and the continuous rule in Eq. (1) on the resistances of the voltage divider.

becomes smaller, the resistances still get a big contribution to their change in value. In our case, the system gets stuck and oscillates between two configurations, spoiling the training procedure thereafter. Experimentally, the presence of noise enables to get out of these problematic configurations and explore the solution space. Recall that, in this simple case, there is an infinite set of solutions for the values of the two resistances: they only need to satisfy Eq. (4) to be solutions. However, as shown in Fig. 3(a) in the paper, the presence of noise only allows the selection of a few favourable, randomly picked configurations in which the MSE decreases of a few orders of magnitude, but there is no overall improvement in the training process.

Therefore, we decide to compare the discrete rule of Eq. (2) with the continuous rule of Eq. (1), the result is shown in **??**.

We can now compare the discrete and continuous rule by plotting the MSE that results from both training methods, this is shown in Fig. 7. We can see how the implementation of the continuous rule enables to reach configurations with a MSE many orders of magnitude smaller than what obtained implementing the discrete rule. There are some oscillations of the MSE in the continuous rule after it has reached a value of the same order of magnitude as the one possessed by the machine precision.

From the results obtained in this simple system, we decided to pursue the idea of finding a physical system that enables the experimental realization of the continuous version if the updating rule on resistances. In our fist attempt, we will consider ionic channels with memristive properties. But, before introducing such systems, in the next section we perform the same analysis in a more complex network.

2.2 More General: 3-input 3-output Network

Let us consider a network made of 9 nodes connected by 16 resistances. Of these nodes, 3 are source nodes, 3 target nodes and 3 hidden nodes. A similar three-input three-output network was considered for allostery tasks in [2].

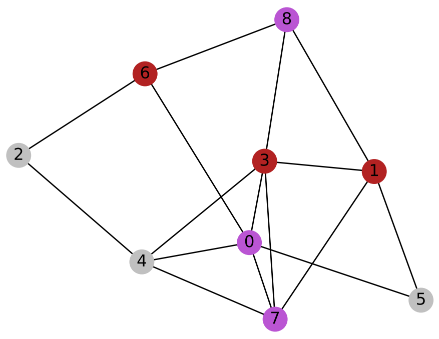

Figure 7: A three-input (red nodes), three-output (purple nodes) network with 16 edges. The allostery task consists in delivering $3V$ in the output nodes, giving 0, 1 and $5V$ to the source nodes. The resistances are initiated to a value of $50k\Omega$.

We first implement the discrete version of the updating rule for resistances and obtain the result shown in Fig. 8. As also discussed in previous work (see [1] Appendix D), learning degrees of freedom in physical systems (in the case here considered, the resistances) share a physical limitation, i.e. their value can not be negative. Therefore, the capacity of the networks considered to learn mappings between sources and targets is limited. Different approaches to circumvent this issue have been explored (works cited in [1], still to be explored), but the authors of the papers we are following simply cut off the values of resistances. In the discrete case, the resistances can assume values between 781Ω and $100k\Omega$.

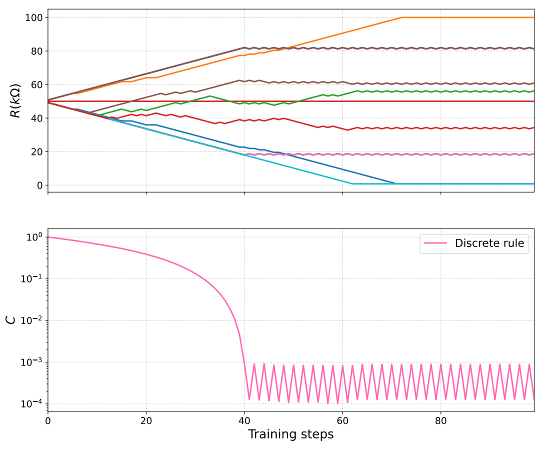

Figure 8: Training the three-input three-output network with the discrete rule for resistances with $\eta = 0.5$.

Comparing the behaviour of the MSE with the one obtained from the implementation of a similar network

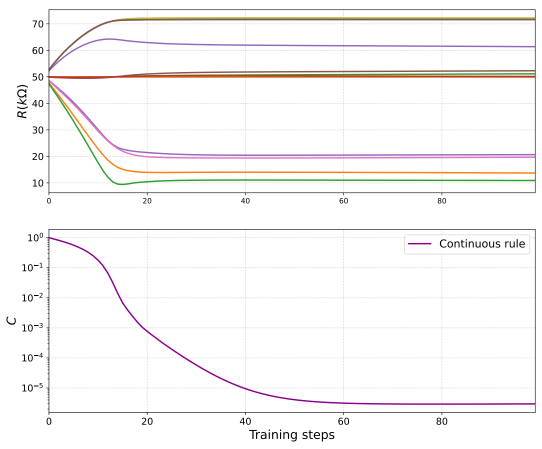

Figure 9: Training the three-input three-output network with the continuous rule for resistances. Here $\eta = 0.5$ and $\alpha = 5 \times 10^3$.

(see Fig. 3(a) in [2]), we find that our training reaches the same order of magnitude as theirs. Recall that we do not implement noise, this forbids getting stuck between two configurations as, on the other hand, happens in our model.

Regarding the choice of the value of the nudging parameter η, from the theory [1] we know that when $\eta \ll 1$, the coupled learning rule mimics the optimization of an effective cost function. On the other hand, the choice of $\eta \sim 1$, as suggested by contrastive Hebbian learning, can affect negatively physical learning, especially in nonlinear networks where the energy landscape in nonconvex and could cause the free and clamped state to reach two different minima, spoiling the training. Therefore, the theory suggests small values of nudging parameter η. However, it was shown [1] that

$$\left[\Delta V^C\right]^2 - \left[\Delta V^F\right]^2 \sim \eta^2$$

which suggests that in an experimental setting when the system is approaching a solution, noise dominates if η is too small. Therefore, the conclusion is that the nudge parameter should be small so that the optimization process gives a good estimate of the desired gradients, but not too small otherwise the learning is dominated by the noise. So far, we have used $\eta = 0.5$ with the sole purpose of replicating the results in the paper and validate our code.

From these results, we can conclude that both the physical limitation on the values of resistances and the discreteness of the updates, still too big when the system approaches a solution, limit the accuracy of the training.

We now switch to the implementation of the continuous rule on the same network. With a fixed value for the nudge parameter $\eta = 0.5$, we still have the freedom to chose the value for the learning parameter α. However, changing the latter only changes of an overall factor the learning rule: the behaviour of the training in time does not change but only the speed is affected. In order to obtain the same time-interval for the training, we set a value of $\alpha = 5 \times 10^3$. The result is shown in Fig. 9.

A comparison between the two rules is now possible, it is shown in Fig. 10. The implementation of the continuous rule results in a better precision of the desired outputs.

However, I am still trying to understand why the the error seems to plateau between 10^{-5} and 10^{-6} instead of keep decreasing. To investigate his point further, I run the algorithm in the continuous setting for more time steps, the result is shown in Fig. 11. We can see that the optimization proceeds and, even if not in a monotonic way, the error decreases of many orders of magnitude. Since the theory suggests a small value of

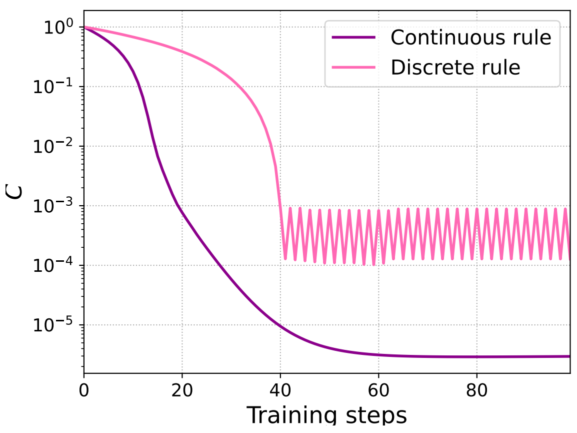

Figure 10: MSE resulting from training a network with a discrete and continuous rule. Here $\eta = 0.5$ and $\alpha = 5 \times 10^3$.

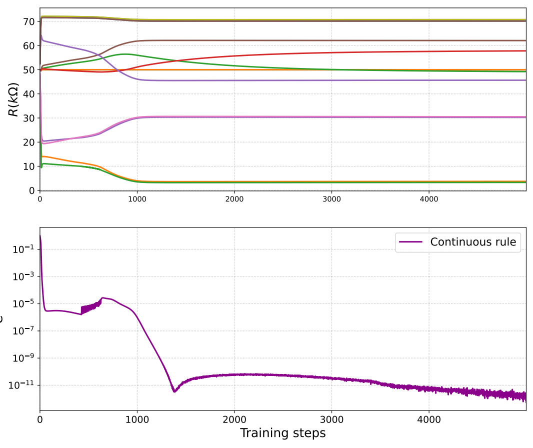

Figure 11: MSE resulting from training a network with a continuous rule for a long period of time. Here $\eta = 0.5$ and $\alpha = 5 \times 10^3$.

nudging parameter η, we compare the training of the continuous rule for different values of this parameter, the result is shown in Fig. 12. We indeed see that a smaller nudging parameter makes training faster and more accurate on the long run.

As a last note, I experimented that the geometry of the network really influences the training. Due to the physical limitation on the values of resistances, many geometries lead to a situation in which, after some of the resistances reach the cut-off, the training is slowed down considerably.

For further developments, it will be necessary to consider methods that avoid going to negative values of the physical quantities of edges. Moreover, it would be interesting to understand better the role of the geometry on the training. To do this, we should first introduce a metric in the network and then study the behaviour of the training as a function if relative distance between inputs-outputs and as a function of the connectivity. But these two parameters are just an example, we can think of more.

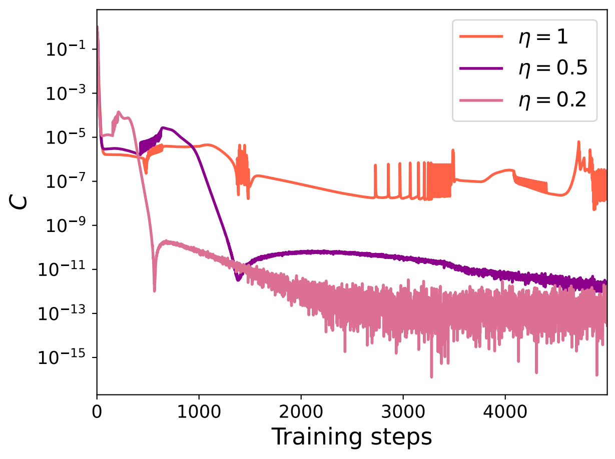

Figure 12: MSE resulting from training a network with a continuous rule for three different values of nudging parameter. Here $\alpha = 5 \times 10^3$.

3 Conical Microfluidic Memristors

The goal of this work is to physically realize a trainable network. We discussed in the last section the limitation of variable resistors, not continuously adjustable, on the accuracy of the output voltages. Moreover, the system needs a supervisor that reads the voltages on output nodes and clamps them depending on the sign of the difference between the voltages in the free and clamped configurations.

On the direction towards a more precise and efficient physical system, we consider resistors that can change their value continuously. Moreover, we would like them to be autonomous, meaning that they don't need the advice of a supervisor. Our first attempt is to substitute resistors with memristors. These devices can change the value of their resistance continuously and have memory, in the sense that the value of the resistance depends on the history of the applied voltage. Perhaps, this property of memory could make the system autonomous.

There are many physically realized memrisotrs, we choose to work with conical channels filled with an aqueous electrolyte since they offer a wide range of achievable values of resistances, they are fast, cheap and easy to fabricate. In previous works [3, 4], these devises were used to perform neuromorphic computing, a computational paradigm that seeks inspiration from the way the brain processes information. They successfully reproduced some neural behaviours, like neural spiking, all-or-none action potentials and short-term plasticity, just to mention some.

3.1 The model

Let us consider a single conical channel, shown in Fig. 13, of length $L = 10\mu$m. It has a base radius of $R_b = 50$nm (radius at $x = 0$), a tip radius (at $x = L$) of $R_t = 50$nm and the radius in between is given by $R(x) = R_b - x\Delta R/L$. The channel connects two reservoirs of electrolyte, where the density of positive and negative ions coincides with the bulk density $\rho_\pm = \rho_b = 0.1$mM. The ions in the electrolyte diffuse with a diffusion coefficient $D = 1.75\mu\text{m}^2\text{ms}^{-1}$. The channel's walls carry a surface charge $e\sigma$ and an electric potential $V(t)$ is imposed on the reservoir, at the base and tip of the channel.

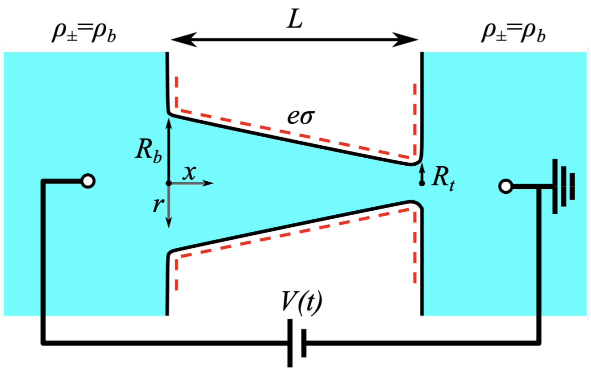

Figure 13: Schematic representation of the conical channel we consider [3].

Transport in the channel is described by the PNPS equations, which were solved for this system in the stationary state version, i.e. with a static external potential V [5]. This results in a static charge current

$$I = g_\infty(V)V$$

where the static conductivity $g_\infty(V)$ was found to be governed by the voltage dependent radially averaged salt concentration $\bar{\rho}_s(x, V)$, here the salt concentration is $\rho_s = \rho_+ + \rho_-$.

For small values of potential V, the average salt concentration in the cone equals the concentration in the bulk $\bar{\rho}_s(x,V) = 2\rho_b$ and the conductance equals the Ohmic conductance

$$g_0 = \frac{\pi R_t R_b}{L}\frac{2\rho_b e^2 D}{k_B T}$$

where $k_B = 1.38 \times 10^{-23}\mathrm{JK}^{-1}$ is the Boltzmann constant and $T = 293.15\mathrm{K}$ the temperature. The resulting current follows $I = g_0 V$. For large static potential drops, the cone shows diodic behaviour as a consequence of the ion concentration polarization. This means that the current can flow through the cone in only one direction, due to the fact that the ions either accumulate or deplete in the channel, depending on the sign of the potential. In this regime, the cone conductance is found by solving the following

$$\frac{g_\infty(V)}{g_0} = 1 + \Delta g \int_0^L \left[\frac{xR_t}{LR(x)} - \frac{e^{\mathrm{Pe}(V)\frac{xR_t^2}{LR_bR(x)}} - 1}{e^{\mathrm{Pe}(V)\frac{R_t}{R_b}} - 1} \right] dx/L \tag{5}$$

where Pe is the Péclet number, for the parameters chosen here $\mathrm{Pe}(V)/V = 16.5V^{-1}$. We show in Fig. 15 the steady state solution for a triangular shaped potential. A positive (negative) potential induces ion deplition (accumulation), thus low (high) conductivity, Fig. 15(a). This gives rise to current rectification, shown in Fig. 15(b) through the relation $I(V) = g_\infty(V)V$.

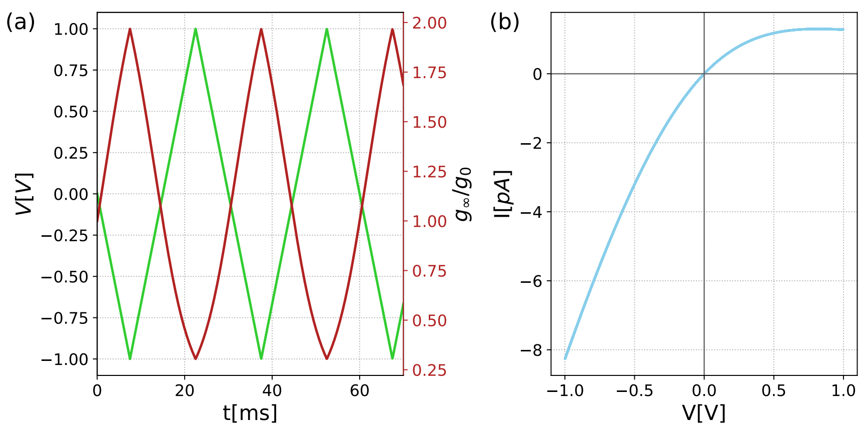

Figure 14: (a) In green the shape of the potential and in red the stationary solution found by solving Eq. (5). In (b) the current found by $I(V) = g_\infty(V)V$.

It was shown that the process of accumulation and depletion occurs over a diffusion-like timescale [6]. For the choice of parameters we made, this corresponds to $\tau = 4.8\mathrm{ms}$.

We now decide to apply a time-dependent potential $V(t)$ and solve of the dynamics of the system. The time-dependent conductance $g(V(t),t)$ is analytically found by assuming an exponential relaxation of the salt concentration with timescale τ towards the steady-state, where the conductance is $g_\infty(V(t))$. This approach leads to the following differential equation for the conductance

$$\frac{\partial g(V(t),t)}{\partial t} = \frac{g_\infty(V(t)) - g(V(t),t)}{\tau} \tag{6}$$

which shows that the **conductance** $g(V(t),t)$ **depends on the entire timetrace of the potential.** The current is then found

$$I(V(t),t) = g(V(t),t)V(t).$$

We show the result of the implementation in Fig. 15. The hysteresis loop in both conductance and current is the hallmark of a memristor.

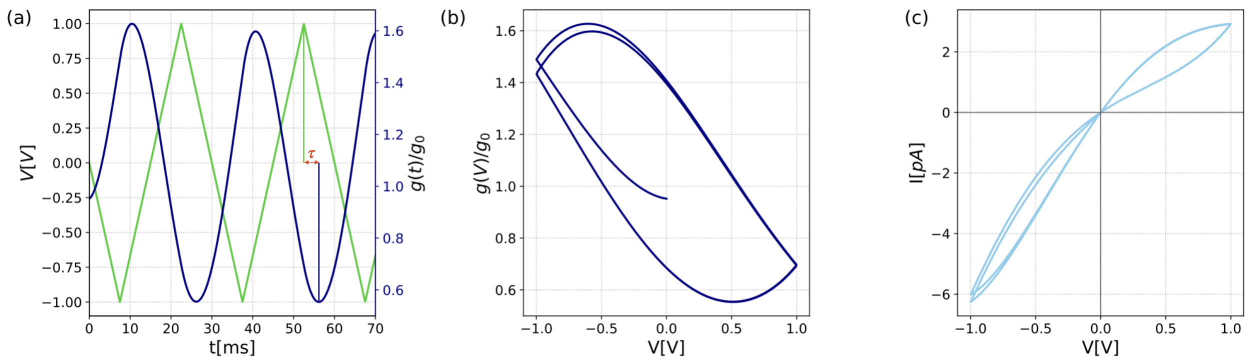

Figure 15: (a) In green the shape of the external potential applied and in blue the conductance of the cone in time. We show the relaxation time τ in red. (b) Conductance as a function of external potential. (c) Current as a function of external potential.

3.2 Simple Start: Voltage Divider

The idea is now to apply the dynamics of memristors to the resistors in the network and find a smart potential that, applied at the edges of the resistors, would mimic the coupled learning training. Once again, we start with the simple voltage divider.

It is necessary to first adjust the system discussed in the previous sections in order to be able to compare its dynamics with the dynamics of memrisors at the same scales. In fact, we want to work with conductances instead of resistances and we want to scale their value from S to the order of pS. Also, we need to go from time-steps to real-time, associating a value of real-time to a single time-step. In Fig. 16 we show the training of conductances with continuous rule of Eq. (3) for the voltage divider.

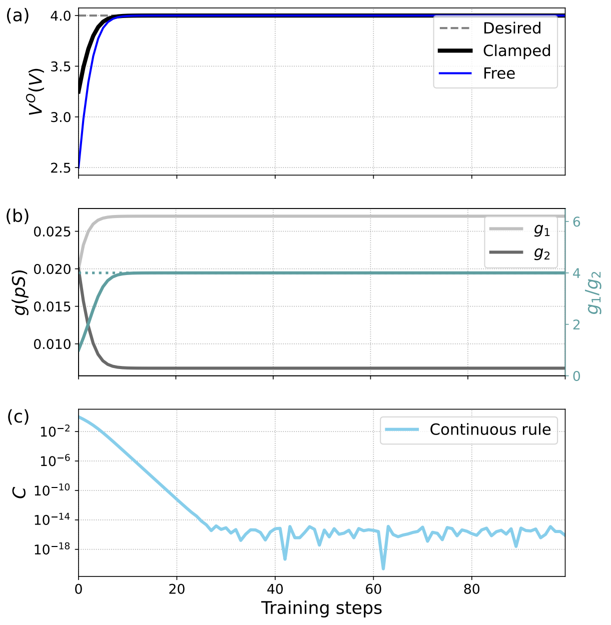

Figure 16: Training a voltage divide within the coupled learning framework, but considering the conductances as learning degrees of freedom instead. Here we used $\eta = 0.5$ and $\alpha = 0.003$.

We started with an intial value for conductances of $1/50$S and adjusted the value of α in order to resemble the behaviour of the resistance, in Fig. 9. Recall that the updating rule of the resistances, Eq. (1), contains resistance squared in the denominator. Therefore, the magnitude and the value of the learning parameter γ is different in the two cases, and the behaviour is not identical.

Before associating a real-time value to the training time-steps, we want to try changing the potential shape imposed at the edges of a conical channel, as illustrated in Fig. 17, and look at the behaviour of the conductance. In the end, we would like to obtain a behaviour similar to the one in Fig. 16(b).

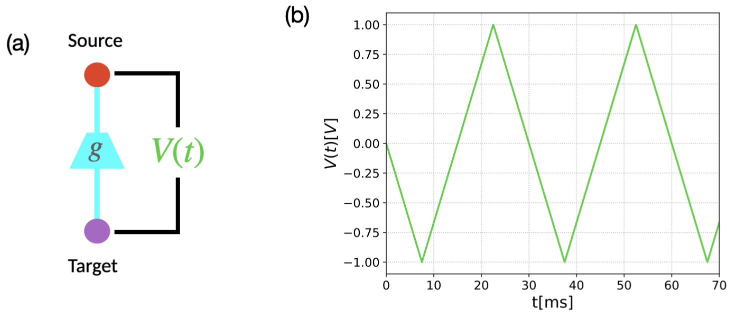

Figure 17: (a) The resistance has been substituted with a conical channel with conductance g. We play with the external potential applied, an example is showed in (b).

Let us first consider the edge relative to the conductance g_1. It connects the source voltage, with $V_1 = 5V$, and the target voltage, in which we would like to have $V_D = 4V$. We first try to impose a potential shaped as a block function that, starting from $5V$ gives pulses to the desired potential of $4V$. We introduce the parameters used to model the square function, it has the equation

$$V(t) = A\,\text{signal.square}\,(2\pi f\, t + \phi, \text{duty}) + V_{\text{shift}}$$

where A is the amplitude of the signal, f the frequency, ϕ the phase and V_{shift} a shift along the y-axis. This signal, in one period $T = 1/f$, assumes two values: $+V$ for $\delta t = T \cdot \text{duty}$ and $-V$, for $\Delta t = T - \delta t$. In Fig. 18 we show the behaviour of the conductance for this type of potential shape. In order for the conductance

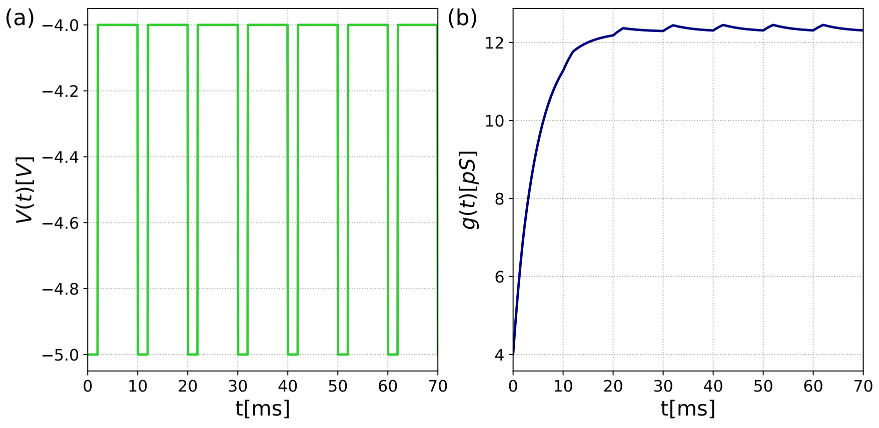

Figure 18: (a) Potential shape: square wave with parameters $A = -1/2$, $f = 0.1$, $\phi = 0$, duty$= 0.2$ and $V_{\text{shift}} = -4.5$. With these parameters the pulses to the desired value of voltage $V_D = 4$ last $\Delta t = 8$ms. (b) Conductance of the conical channel in time when external voltage in (a) is applied. The parameters have not changed, therefore $\tau = 4.8$ms.

to grow, we need to impose a negative voltage. I observed that changing the initial condition $g(0)$ does not change the behaviour at longer times, the conductance always tends to the final value of approximately $12.3p$S. Changing Δt only changes the length of the oscillations at longer times.

With this, we build a voltage divider that has in one edge (g_1) a memristor and in the other (g_2) a resistor trained with coupled learning. The result in shown in Fig. 19. We imposed the initial condition for the conductances to be $g(t=0) = 4p$S, changing its value doesn't affect much the result. We perform 1000 time-steps in both trainings. In order to have both trainings in real time, we associate a value of 0.07ms to a training step in the coupled learning framework. It seems that the memristor follows its dynamic independently,

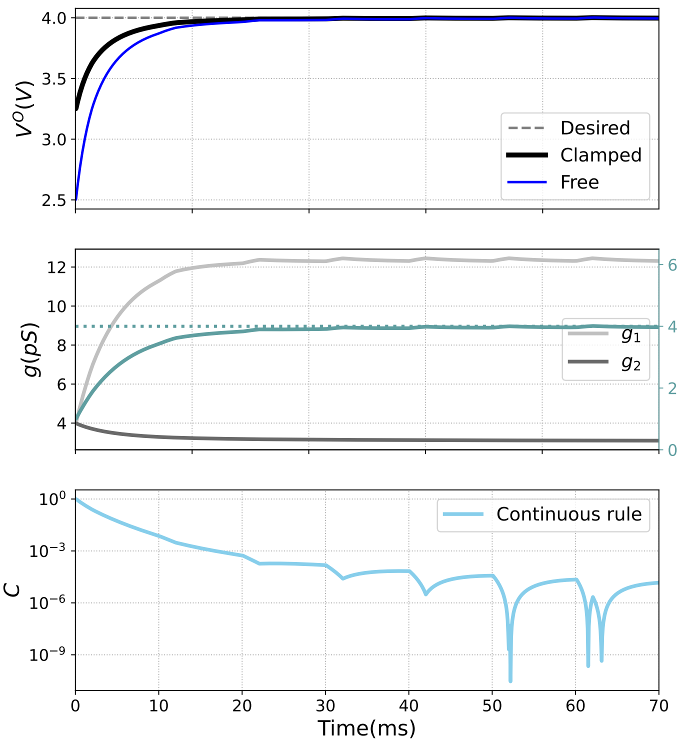

Figure 19: Training the system with one memeristor (g_1) that follows the dynamics in Eq. (6), and a resistor (g_2) that follows coupled learning method.

while the resistor, with coupled learning, tries to adjust accordingly.

Regarding the memristor g_2, we apply another square wave potential, this time oscillating between the source potential $0V$ and the desired $4V$. In Fig. 20 we show the result on the behaviour of the conductance. Due to the larger difference between the values of the voltages with respect to the voltages in g_1, the oscillations at larger times are more marked.

We now try to substitute both the edges with memresistors. On g_1 we apply an external potential as in Fig. 18, while on g_2 as in Fig. 20. The result is shown in Fig. 21. The network this way does not get trained. This is plausable, the updating this way is independent of the state of the system and we are not using the memory properties of the memristor.

With this machinery now working, we want to find a smart way of utilizing the memristors, at their full capacity.

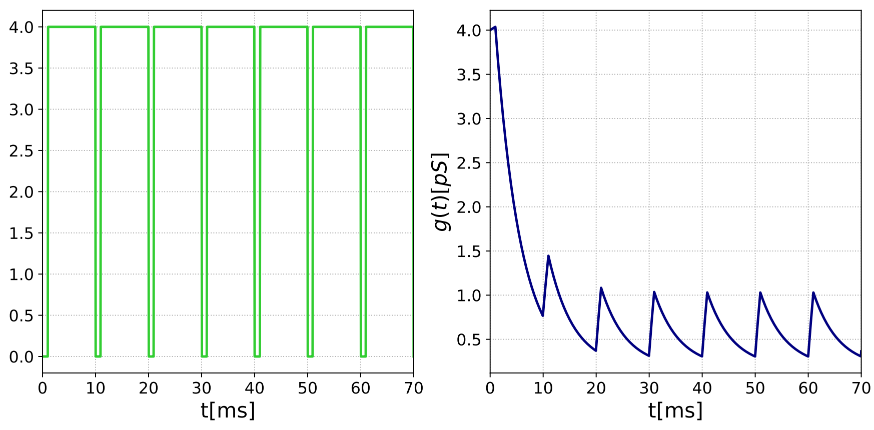

Figure 20: (a) Potential shape: square wave with parameters $A = -1/2$, $f = 0.1$, $\phi = 0$, duty= 0.2 and $V_{\text{shift}} = -4.5$. With these parameters the pulses to the desired value of voltage $V_D = 4$ last $\Delta t = 8$ms. (b) Conductance of the conical channel in time when external voltage in (a) is applied. The parameters have not changed, therefore $\tau = 4.8$ms.

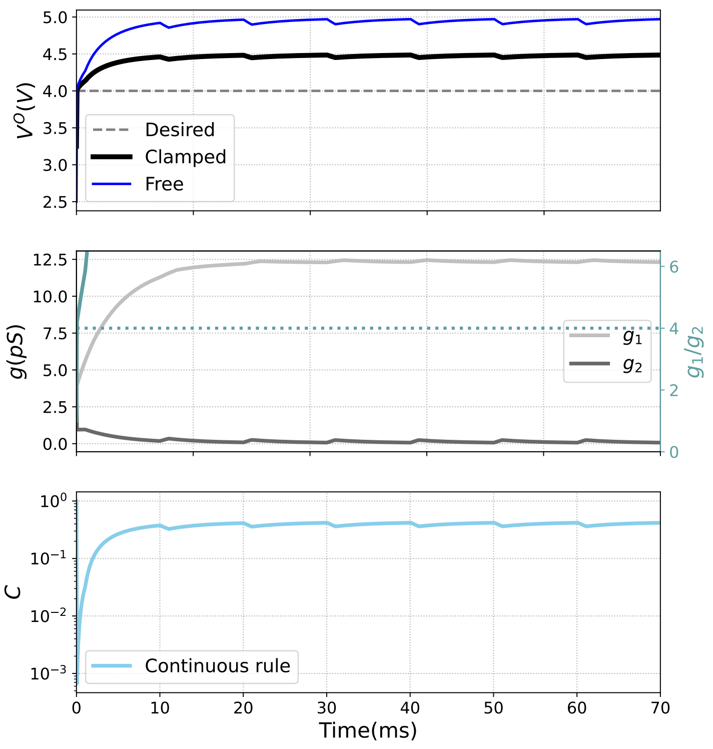

Figure 21: Training a voltage divider with two conical microfluidic channels connecting the nodes.

References

[1] Menachem Stern, Daniel Hexner, Jason W. Rocks, and Andrea J. Liu. Supervised learning in physical networks: From machine learning to learning machines. *Physical Review X*, 11(2), May 2021.

[2] Sam Dillavou, Menachem Stern, Andrea J. Liu, and Douglas J. Durian. Demonstration of decentralized physics-driven learning. *Physical Review Applied*, 18(1), July 2022.

[3] T.M. Kamsma, W.Q. Boon, T. ter Rele, C. Spitoni, and R. van Roij. Iontronic neuromorphic signaling with conical microfluidic memristors. *Physical Review Letters*, 130(26), June 2023.

[4] T. M. Kamsma, J. Kim, K. Kim, W. Q. Boon, C. Spitoni, J. Park, and R. van Roij. Brain-inspired computing with fluidic iontronic nanochannels, 2023.

[5] Willem Q. Boon, Tim E. Veenstra, Marjolein Dijkstra, and René van Roij. Pressure-sensitive ion conduction in a conical channel: Optimal pressure and geometry. *Physics of Fluids*, 34(10), October 2022.

[6] Dengchao Wang and Gangli Wang. Dynamics of ion transport and electric double layer in single conical nanopores. *Journal of Electroanalytical Chemistry*, 779:39–46, 2016. Special issue in honor of Koichi Aoki.