# Appendix

## A Proof of Theorem 1

In this section, we provide the proof to Theorem 1 given in our letter for general $t \in [t_k, t_{k+1})$, $k \in \mathbb{Z}$. As a reminder, the initial-value-problem (IVP) of a liquid-time-constant (LTC) neuron is stated below by:

$$\frac{\mathrm{d}}{\mathrm{dt}}x(t) = -\omega x(t) + \sum_{s=1}^{S} f_s(\tilde{g}_s(t), \boldsymbol{\theta_s})(A_s - x(t)), \tag{1}$$

with presynaptic signals

$$\tilde{g}(t) = \sum_{k \in \mathbb{Z}_+} c[k]\beta_k(t). \tag{2}$$

Throughout our manuscript, we use square brackets, $[\cdot]$, for discrete sequences and round brackets, $(\cdot)$, for continuous functions.

*Theorem 1:* Given an LTC neuron defined by (1), with $S$ presynaptic input signals $\tilde{g}_s(t)$ as described in (2), its state $x$ at time $t = t_k$ can be evaluated as:

$$x(t)|_{t=t_k} = \frac{1}{\alpha[t_k]}\left(x(0) + \sum_{i=0}^{k-1}(\alpha[t_{i+1}] - \alpha[t_i])\sum_{s=1}^{S} A_s u_s[t_i]\right), \tag{3}$$

where

$$\alpha[t_k] = e^{\omega t_k + \sum_{i=0}^{k-1}\sum_{s=1}^{S}(t_{i+1}-t_i)f_s(c_s[i])},$$
$$u_s[t_i] = \frac{f_s(c_s[i])}{\omega + \sum_{r=1}^{S} f_r(c_r[i])}.$$

*Proof*

Let us denote

$$\eta(t) = \sum_{s=1}^{S} f_s(\tilde{g}_s(t)), \tag{4}$$

$$\xi(t) = \sum_{s=1}^{S} A_s f_s(\tilde{g}_s(t)). \tag{5}$$

The equivalent problem to the ODE in (1) is

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = -(\omega + \eta(t))x(t) + \xi(t). \tag{6}$$

Let us define for piecewise constant inputs and $t \in [t_k, t_{k+1})$

$$\begin{aligned}
\alpha(t) &= e^{\int_0^t (\omega+\eta(u))du} \\
&= e^{\omega t + \int_0^t \eta(u)du} \\
&= e^{\omega t + \int_0^t \sum_{s=1}^S f_s(\tilde{g}_s(u))du} \\
&= e^{\omega t + \sum_{s=1}^S \int_{t_k}^t f_s(\tilde{g}_s(u)))du + \sum_{s=1}^S \sum_{i=0}^{k-1} \int_{t_i}^{t_{i+1}} f_s(\tilde{g}_s(u))du} \\
&= e^{\omega t + \sum_{s=1}^S (t-t_k) f_s(c_s[k]) + \sum_{s=1}^S \sum_{i=0}^{k-1} (t_{i+1}-t_i) f_s(c_s[i])}
\end{aligned} \tag{7}$$

By multiplying both sides of (6) by $\alpha$, we obtain:

$$\alpha(t)\frac{\mathrm{d}}{\mathrm{d}t}x(t) = -(\omega + \eta(t))\alpha(t)x(t) + \xi(t)\alpha(t). \tag{8}$$

When taking the derivative of $\alpha(t)x(t)$, we observe that

$$\alpha(t)\frac{\mathrm{d}}{\mathrm{d}t}x(t) + (\omega + \eta(t))\alpha(t)x(t) = \frac{\mathrm{d}}{\mathrm{d}t}\alpha(t)x(t). \tag{9}$$

Inserting (9) into (8), we have

$$\alpha(t)x(t) = \int_0^t \xi(u)\alpha(u)du + C. \tag{10}$$

First, let us find $C$:

$$\alpha(t=0) = 1 \quad \Rightarrow \quad C = x(t=0) \tag{11}$$

$$\alpha(t)x(t) - x(0) = \int_0^t \xi(u)\alpha(u)du \tag{12}$$

Let us now compute $\int_0^t \xi(u)\alpha(u)du$:

$$\begin{aligned}
\int_0^t \xi(u)\alpha(u)du &= \int_0^t \sum_{s=1}^S A_s f_s(g_s(u))\alpha(u)du \\
&= \sum_{s=1}^S A_s \int_0^t f_s(g_s(u))\alpha(u)du \\
&= \sum_{s=1}^S A_s I(t,s),
\end{aligned} \tag{13}$$

where we define $I(t,s) = \int_0^t f_s(\tilde{g}_s(u))\alpha(u)du$. This expression, for piecewise constant inputs, becomes

$$\begin{aligned} I(t,s) &= \int_0^t f_s(\tilde{g}_s(u))\alpha(u)du \\ &= \int_{t_k}^t f_s(\tilde{g}_s(u))\alpha(u)du + \sum_{i=0}^{k-1}\int_{t_i}^{t_{i+1}} f_s(\tilde{g}_s(u))\alpha(u)du \\ &= \int_{t_k}^t f_s(\tilde{g}_s(u))e^{\omega u+\int_0^u \eta(v)dv}du + \sum_{i=0}^{k-1}\int_{t_i}^{t_{i+1}} f_s(\tilde{g}_s(u))e^{\omega u+\int_0^u \eta(v)dv}du. \end{aligned} \tag{14}$$

As $u \in [t_i, t_{i+1})$, we have

$$\begin{aligned} I(t,s) = &\int_{t_k}^t f_s(c_s[k])e^{\omega u+\sum_{r=1}^S (u-t_k)f_r(c_r[k])+\sum_{r=1}^S\sum_{j=0}^{k-1}(t_{j+1}-t_j)f_r(c_r[j])}du \\ &+ \sum_{i=0}^{k-1}\int_{t_i}^{t_{i+1}} f_s(c_s[i])e^{\omega u+\sum_{r=1}^S (u-t_i)f_r(c_r[i])+\sum_{r=1}^S\sum_{j=0}^{i-1}(t_{j+1}-t_j)f_r(c_r[j])}du. \end{aligned} \tag{15}$$

Using a change of variable with

$$z(u) = \omega u + \sum_{r=1}^S (u-t_i)f_r(c_r[i]) + \sum_{r=1}^S\sum_{j=0}^{i-1}(t_{j+1}-t_j)f_r(c_r[j]) \tag{16}$$

where

$$\frac{\mathrm{d}}{\mathrm{d}u}z(u) = \omega + \sum_{r=1}^S f_r(c_r[k]), \tag{17}$$

we obtain

$$I(t,s) = f_s(c_s[k])\frac{e^{\omega t+\int_0^t \eta(u)du} - e^{\omega t_k+\int_0^{t_k}\eta(u)du}}{\omega+\sum_{r=1}^S f_r(c_r[k])} + \sum_{i=0}^{k-1} f_s(c_s[i])\frac{e^{\omega t_{i+1}+\int_0^{t_{i+1}}\eta(u)du} - e^{\omega t_i+\int_0^{t_i}\eta(u)du}}{\omega+\sum_{r=1}^S f_r(c_r[i])}. \tag{18}$$

Then, from (10), the final expression for the neuron's dynamic is the following:

$$\begin{aligned} x(t) = \frac{1}{\alpha(t)}\Bigg(x(0) + \sum_{s=1}^S A_s \Bigg( &f_s(c_s[k])\frac{e^{\omega t+\int_0^t \eta(u)\,du} - e^{\omega t_k+\int_0^{t_k}\eta(u)\,du}}{\omega+\sum_{r=1}^S f_r(c_r[k])} \\ &+ \sum_{i=0}^{k-1} f_s(c_s[i])\frac{e^{\omega t_{i+1}+\int_0^{t_{i+1}}\eta(u)\,du} - e^{\omega t_i+\int_0^{t_i}\eta(u)\,du}}{\omega+\sum_{r=1}^S f_r(c_r[i])}\Bigg)\Bigg) \end{aligned} \tag{19}$$

Rewriting (10) with the expressions of $\alpha$ (7) and $\eta$ (4):

$$x(t) = \frac{1}{\alpha(t)}\left(x(0) + \sum_{s=1}^S A_s f_s(c_s[k])\frac{\alpha(t)-\alpha[t_k]}{\omega+\eta[t_k]} + \sum_{s=1}^S\sum_{i=0}^{k-1} A_s f_s(c_s[i])\frac{\alpha[t_{i+1}]-\alpha[t_i]}{\omega+\eta[t_i]}\right). \tag{20}$$

To simplify the given expression for $x(t)$:

$$x(t) = \frac{1}{\alpha(t)} \left( x(0) + (\alpha(t) - \alpha[t_k]) \sum_{s=1}^{S} A_s u_s[t_k] + \sum_{i=0}^{k-1} (\alpha[t_{i+1}] - \alpha[t_i]) \sum_{s=1}^{S} A_s u_s[t_i]) \right). \tag{21}$$

This expression for $x(t)$ for $t = t_k$ simplifies to:

$$x(t)|_{t=t_k} = \frac{1}{\alpha[t_k]} \left( x(0) + \sum_{i=0}^{k-1} (\alpha[t_{i+1}] - \alpha[t_i]) \sum_{s=1}^{S} A_s u_s[t_i] \right). \tag{22}$$

□

# B Proof of Corollary 1

*Corollary 1:* For $t_k \leq t_1 < t_{k+1} \leq t_2 < t_{k+2}$ and $t_1, t_2 \in \mathbb{R}, k \in \mathbb{N}$, the state of the LTC neuron at $t_2 = t_{k+1}$ is given recursively as:

$$x[t_{k+1}] = \gamma[t_k] \left( x[t_k] - \sum_{s=1}^{S} A_s u_s[t_k] \right) + \sum_{s=1}^{S} A_s u_s[t_k] \tag{23}$$

where

$$\gamma[t_k] = e^{-(t_{k+1} - t_k)(\omega + \sum_{s=1}^{S} f_s(c_s[k]))}.$$

*Proof*

By replacing $t = t_{k+1}$ in (22):

$$\begin{aligned} x[t_{k+1}] &= \alpha^{-1}[t_{k+1}] \left( x(0) + \sum_{i=0}^{k} (\alpha[t_{i+1}] - \alpha[t_i]) \sum_{s=1}^{S} A_s u_s[t_i] \right) \\ &= \alpha^{-1}[t_{k+1}] \left( \alpha[t_k] x[t_k] + (\alpha[t_{k+1}] - \alpha[t_k]) \sum_{s=1}^{S} A_s u_s[t_k] \right) \\ &= \alpha^{-1}[t_{k+1}] \alpha[t_k] \left( x[t_k] - \sum_{s=1}^{S} A_s u_s[t_k] \right) + \sum_{s=1}^{S} A_s u_s[t_k] \end{aligned}$$

□

# C Details of experiments

All experiments were conducted on an Apple M3 Mac with 128 GB of memory and 40 GPU cores. All networks were trained for maximum 100 epochs, with early stopping enabled and a learning rate scheduler.

## C.1 Datasets

**MNIST:** We create a sequential image processing task using the handwritten digits from the MNIST dataset. To this end, each $28 \times 28$ image from the datasets is converted into a 1-dimensional sequence by concatenating its rows. The network architecture consists of one layer with 8 neurons, followed by batch normalization, flattening, and a dense layer with softmax activation. The split between training, validation and out-of-sample test set is 48,000, 12,000 and 10,000.

**Human Acitivity Recognition (HAR):** The dataset consists of recordings of 30 subjects performing six different activities (*e.g.* walking, sitting, laying) while wearing smartphones equipped with embedded sensors. Following the same preprocessing step as in [1], the dataset results in a total of 826 sequences with 561 features at each timestep. We classify each activity using a network with one layer of 8 neurons and a time-distributed dense layer.

**MIT-BIH Arrythmia (ECG):** This dataset contains 48 hours of ECG recordings from patients, sampled at 360 Hz. The goal is to classify them into five categories of heartbeats. The dataset is split into training, test and validation sets of 70043, 21892 and 17511 timesteps. The network is designed to first process the inputs with a convolutional block consisting of three convolutional layers with 64 filters each, then pass the signal into a LTC layer of 64 neurons followed by batch normalization, ReLU, average pooling and flattening, and finally apply a dense layer.

**Person Activity Recognition (PAR):** The data consists of recordings from four sensors for five persons, modified to have seven different categories of acitvies. Each input timestep consists of seven dimensions: four representing the sensor IDs and three for the sensor values. The data is split into training, test and validation sets of size 6993, 776 and 1942 sequences. We used the same preprocessing as described in [1] to represent the sensor data as an irregularly sampled time series. The model processes the features and the timesteps using an LTC layer with 64 neurons, and outputs the corresponding category with a time-distributed dense layer.

## C.2 Network training parameters

The parameters used in all experiments are summarized in Table 1. To better understand the differences in the backbone ODE of the LTC neuron, we used the Cf-S variant from the CfC architecture [2] ("CfC"), which represents a closed-form solution network directly derived from the IVP solution. For all experiments, "CfC" was configured with one backbone layer of 128 units, LeCun tanh activation, and a dropout rate of 0.1. The "Euler" layer approximates the IVP defined in (1) iteratively, using explicit Euler integration with multiple time steps. In contrast to the Exact solution where we tested different presynaptic nonlinearities f (such as sigmoid, ReLU or ReLU activated dense layer). The Euler layer was only evaluated with sigmoidal activation. However, as shown in Table 3 in the manuscript, the "Exact-Sigmoid" solution already achieves greater accuracy than the "Euler" layer, making further testing of additional variants of the "Euler" layer unnecessary. It is important to emphasize that the objective of our experiments was not to design the best predictive models, but rather to empirically assess and compare the expressive power of different ODE solutions.

Table 1: Training parameters used during training for neural network experiments.

| Parameters | Layer | | | | |
|---|---|---|---|---|---|
| | Euler | Cf-S | Exact-Sigmoid | Exact-ReLU | Exact-Dense |
| optimizer | Adam | Adam | Adam | Adam | Adam |
| batch size | 64 | 64 | 64 | 64 | 64 |
| epochs | 100 | 100 | 100 | 100 | 100 |
| learning rate | 0.05 | 0.001 | 0.05 | 0.05 | 0.005 |

# References

[1] R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, "Liquid time-constant networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, pp. 7657–7666, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16936

[2] R. Hasani, M. Lechner, A. Amini, L. Liebenwein, A. Ray, M. Tschaikowski, G. Teschl, and D. Rus, "Closed-form continuous-time neural networks," *Nature Machine Intelligence*, vol. 4, pp. 1–12, 11 2022.