UPPSALA
UNIVERSITET

1FA492 13164

Project 15 hp

Dec 2025

Neuromorphic Circuit Design

Neuromorphic circuit design and simulation based on ferroelectric and antiferroelectric transistors

Authors: Alexander Karlsson, Hannes Tjulin

Supervisor: Libo Chen

Master's Programme in Engineering Physics

UPPSALA
UNIVERSITET

Neuromorphic Circuit Design

Alexander Karlsson, Hannes Tjulin

Abstract

The recent explosion of commercially available artificial intelligence (AI) has led to an increase in power consumption, which is expected to continue. This is in part due to the used hardware, which is inefficient in mimicking the dynamics of biological brains. Therefore, research is being done in new hardware using new materials, which could potentially drastically reduce the power consumption of AI.

A circuit has been proposed that should mirror the dynamics of neurons. A key component is a new ”emerging device” which exhibits hysterectic properties: its output signal is dependent on the instantaneous input signal, but also on the history of input signals the device has received. The goal of this project is to theoretically determine ideal parameter values of this hysteresis device.

To achieve this, a mathematical model of the artificial neuron has been developed and implemented in a simulation. This artificial neuron is shown to be able to replicate realistic neuron behavior such as adaptability. Causes of failure have been identified: mainly latch-up where the circuit gets stuck in a state and is unable to fire output spikes. In addition, the dependence of output frequency on different parameters has been investigated. These results show that this artificial neuron is a plausible candidate for a low-power implementation of AI hardware, and paves the way for research in which materials could have ideal properties.

Faculty of Science and Technology

Uppsala university

Supervisor: Libo Chen

Table of Contents

1 Introduction

The recent revolution in artificial intelligence (AI) cannot have escaped anyone's notice; best exemplified in the advances made in large language models such as ChatGPT, the research field was honored with a Nobel Prize in 2024 [1]. This boom is however not without consequences, as it has led to a vast increase in energy consumption for the AI data centers. The International Energy Agency (IEA) for example predicts that the global energy usage of data centers will increase by 15% annually until 2030, landing at a 3% share of global energy consumption [2]. A large part of this increase is driven by the evolution of AI.

This fact has driven research into searching for more efficient hardware for AI computation. One path for optimization is neuromorphic computing, where one tries to mirror the speed and energy efficiency of biological brains. There are currently neuromorphic devices on the market, such as IBM's TrueNorth chip [3] and Intel's Loihi chip [4]. These solutions rely on CMOS (complementary metal oxide semiconductor) technology, which is well-tested and easily scalable in production. It has some drawbacks however: the energy consumption is still many times higher than a biological brain, in no small part due to the need of additional circuitry to replicate advanced neuron behavior that is not reflected in CMOS material properties [5]. A path forward is therefore to develop artificial neurons with new materials, whose intrinsic properties can mimic neuron behavior well.

1.1 Problem Description

To realize a more energy efficient hardware for AI, in particular neural networks, a circuit mirroring the function of the brain has been proposed by Libo Chen[1], supervisor of this project. The idea of this circuit is to in response to an external stimulus (voltage), reaching a certain threshold, a neuron triggers and outputs a voltage spike. Thus, sending in a constant voltage, the neuron should accumulate charge and release spikes with a certain frequency until the voltage is turned off. What is new with this idea is a component in the circuit which has intrinsic hysterectic properties (e.g. an anti-ferroelectric material). This should allow for adaptation to stimuli by increasing the threshold for the neuron after each spike, consequently increasing the time between spikes for a constant stimulus. This added threshold should then decay back to the original after a certain decay time. These artificial neurons can then be connected to create a hardware version of a neural network that is more energy efficient than running neural network software on logic circuits.

This circuit can be built by using an emerging device with a hysteresis like behavior in combination with a MOSFET and a RC circuit, which will be more thoroughly explained in Section 2. The idea behind this circuit is that a incoming voltage charges up the capacitor, which then reaches a threshold voltage where the device outputs a voltage connected to the MOSFET gate, which then discharges the capacitor until the device returns to it's starting voltage and a new spike can be generated. Thus when applying a constant voltage, the output will be in the form of spikes generated by charging and discharging the capacitor.

The purpose of this project is to create a simulation of this device and circuit in python by setting up the equations of the circuit and iteratively solving them as time progresses. Furthermore, the purpose includes observing the necessary requirements for the parameters to optimize the energy consumption while keeping the parameters in a realistic range. It is also important to gain an understanding of how the parameters relate to each other and the stability of the simulation so that device specifications can be extracted for further work.

[1] Libo Chen, Uppsala University: Assistant Professor at Department of Electrical Engineering; Solid-State Electronics

2 Theory

In the following section, the theory underpinning this project is described. To begin with, the theory behind this specific neuron model and its main component is described. After that, the scope is widened to discuss the desired behavior of an artificial neuron, and which characteristics of a biological neuron it should be able to mimic. Finally, a way to create a more advanced neuron circuit model is presented, followed by an example of neuromorphic computing already commercially available is given.

2.1 Hysteresis Device

The main component of the proposed neuron circuit is the so-called hysteresis device, whose I-V characteristics are in the form of a hysteresis curve. A practical implementation could for example be an anti-ferroelectric neuron [6], but this work looks for an ideally fabricated hypothetical device which need not be based on the properties of a specific material. An ideal hysteresis curve with relevant parameters is shown to the left in Figure 1.

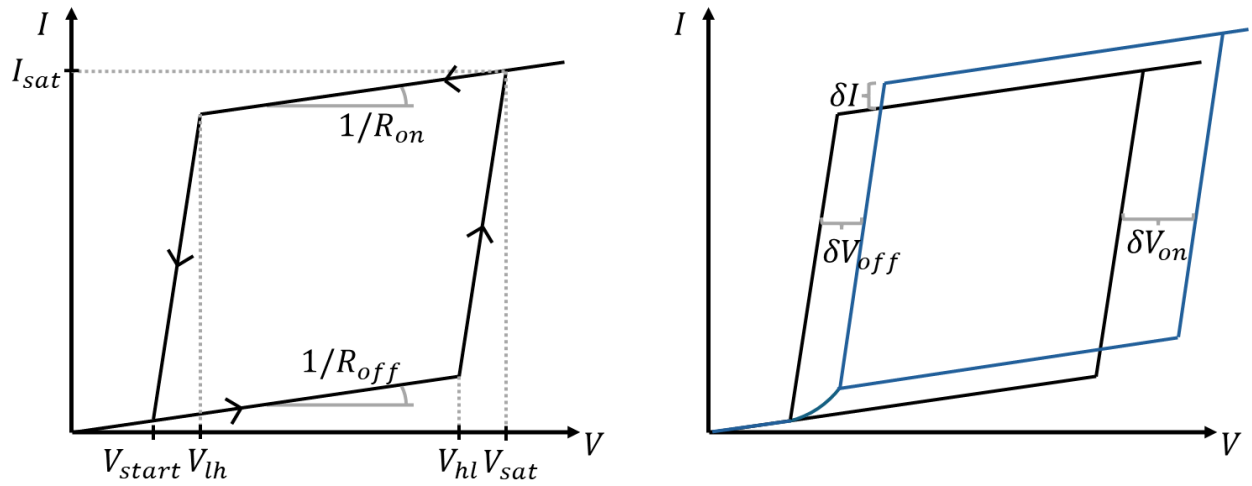

Figure 1: Hysteresis loops with parameters written out. Left: One loop with default parameters shown. Right: after one completed loop, the second loop is shifted and connected smoothly to the old V_{start}

The two marked slopes are given as $1/R$ to use the language of an electric circuit, connecting I and V according to $I = V/R$. A distinction is made between the two important states, in the off-state (with slope $1/R_{off}$), the output current is negligible compared to in the on-state (with slope $1/R_{on}$). The nomenclature V_{hl} and V_{lh} comes from the switching of states: the voltage where the component goes from high to low ”resistance” (meaning low to high current) or vice versa.

After each completed loop, the properties of the device change slightly. To the right in Figure 1, the parameters describing these shifts are shown. All slopes are preserved, but all points are shifted by some amount δx. All points are shifted upwards by δI, but the points to the left and the right of the loop can be shifted in the V-dimension by different amounts. This shift is not constant, but decays exponentially from its maximum value δx_0:

$$\delta x(t) = \delta x_0 e^{-t/\tau} \tag{1}$$

The decay rate τ is for simplicity chosen to be the same for all δx. This means that if the time between loops is long enough (in comparison to the decay rate τ), two subsequent loops should look identical.

2.2 Circuit Design

The layout of a circuit modeling the LIF-neuron is found in Figure 2. The most important blocks of the circuit consists of the resistance-capacitor (RC) sub-circuit which accumulates charge when a voltage difference (V_{in}) is applied over R_{in} and C. This charge then slowly leaks over R_L. Another sub-circuit is the hysteresis device and nMOSFET, where the nMOSFET more or less allows a current through when a gate-source voltage is above a certain threshold, V_{th}. Thus this sub-circuit allows current to pass the nMOSFET when the hysteresis device is on. The general behavior of the circuit can thus be described as follows:

1. Upon application of an input voltage, the RC sub-circuit consisting of R_{in} and C charges up, leading to V_{mem} increasing.
2. The charge on C can dissipate through the leak resistance R_L.
3. If V_{mem} reaches the V_{hl} of the hysteresis device, the latter turns on, giving a sharp increase in V_{out}.
4. A high V_{out} means a high gate voltage on the MOSFET, allowing a current, I_{DS}, to pass through.
5. I_{DS} quickly discharges the capacitor, lowering V_{mem} until it is lower than V_{lh}, turning off the hysteresis device. Consequentially, the MOSFET turns off and we are back at step 1.

From these steps, the following two prerequisites can be identified for a spiking output.

1. $V_{in} > V_{hl}$ is required for the hysteresis device to turn on.
2. $V_{out} > V_{th}$ is required to turn on the MOSFET.

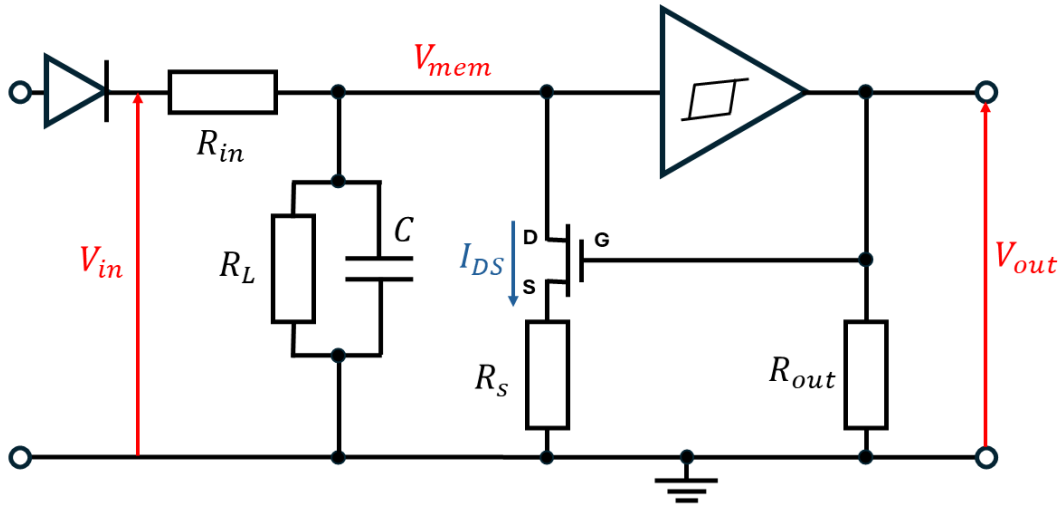

Figure 2: Sketch of the LIF circuit. V_{in} is the input voltage. V_{mem} is the equivalent to the membrane voltage in a biological neuron, and is what activates the spike. C is the capacitance of the capacitor. R_{in}, R_L, R_S and R_{out} are the in-, leak-, source- and out-resistances respectively. Finally, I_{DS} is the current passing through the MOSFET transistor.

The dynamics of the system are described fully by equations 2–6. Their derivation can be found in Appendix A. Equation 2 originates from the RC sub-circuit and refers to the how V_{mem} changes due to the parameters in that subcircuit and the current passing through the nMOSFET. The formula in Equation 3 is for a N-channel MOSFET according to the Shichman-Hodges model [7; 8]. Since both V_{GS} and V_{DS} are arguments of and depend on I_{DS}, the function is recursive and needs to be solved numerically. In essence are these the equations that are simulated.

$$\frac{\mathrm{d}V_{mem}}{\mathrm{d}t} = \begin{cases} \frac{V_{in} - V_{mem}}{R_{in}C} - \frac{I_{DS}}{C} - \frac{V_{mem}}{R_L C}, & V_{in} > V_{mem} \\ -\frac{I_{DS}}{C} - \frac{V_{mem}}{R_L C}, & V_{in} \leq V_{mem} \end{cases} \tag{2}$$

$$I_{DS} = \begin{cases} 0, & V_{GS} < V_{th} \\ K\Big((V_{GS} - V_{th})V_{DS} - V_{DS}^2/2\Big)(1 + \lambda V_{DS}), & 0 < V_{DS} < V_{GS} - V_{th} \\ (K/2)(V_{GS} - V_{th})^2(1 + \lambda V_{DS}), & 0 < V_{GS} - V_{th} < V_{DS} \end{cases} \tag{3}$$

$$V_{GS} = V_{out} - R_D I_{DS} \tag{4}$$

$$V_{DS} = V_{mem} - R_D I_{DS} \tag{5}$$

$$V_{out} = R_{out} \cdot H(V_{mem}, t) \tag{6}$$

Moreover, to get an estimate of the energy each spike carries and the power consumption of the spikes one can assume that during the spike generation (Hysteresis is in ”on” state) the resistance is constant but the voltage

changes linearly with time from V_{sat} to V_{lh} during the spike time, ΔT. Thus one can estimate the energy of each spike with the following formula:

$$P(t) = \frac{V_{out}(t)^2}{R_{on}} \tag{7}$$

$$\implies E_{spike} = \int_0^{\Delta T} \frac{V_{out}(t)^2}{R_{on}} dt = \frac{\Delta T}{R_{on}} \left[V_{out,sat} V_{out,lh} + \frac{(V_{out,lh} - V_{out,sat})^2}{3} \right] \tag{8}$$

$V_{out,sat}$ and $V_{out,lh}$ are both determined by the current generated over the device going through the out-resistance, R_{out}, by $V_{out} = I_{out} R_{out}$

Furthermore, the estimation of the power consumption is made by multiplying the energy per spike with the frequency of the spikes. These values thus give a rough estimation of the energy consumption of the spike generation. However, a more throughout calculation or experimental determination is required for a precise power consumption of the whole circuit including joule heating losses and MOSFET leakage.

2.3 Artificial Neurons

The artificial neuron, mentioned in Section 1, should ideally fulfill six dynamic and three functional key criteria to optimally mimic biological neurons and effectively create a hardware platform for neural networks. These criteria have been identified by the project supervisor, Libo Chen, and are yet to be published, but similar criteria can be found in other publications [9]. The dynamic criteria are:

- **Integration:** When external stimuli (voltage) are applied, the neuron should accumulate charge.
- **Leakage:** When no external stimuli, or a small external stimulus, is applied the neuron's accumulated charge should slowly discharge (Leak).
- **Formal spike generation:** When the accumulated charge reaches a threshold voltage, the neuron should output a voltage spike
- **Resets:** After a spike has been produced, the neuron should reset to it's initial state, thus being able to produce another spike
- **Adaptation:** Under continuous stimuli, the neuron should adapt to the stimuli, decreasing the spiking frequency.
- **Dynamic threshold:** After neuron spiking, the threshold voltage for the next spike should slightly increase. This increase should decay to the original after a specified decay time.

The effect of these dynamic criteria can be seen in Figure 3. An artificial neuron that Leaks, Integrates and Fires (generates spikes) is called a LIF-neuron, and can be considered the simplest neuron model. The remaining dynamic criteria allow for more advanced spiking patterns.

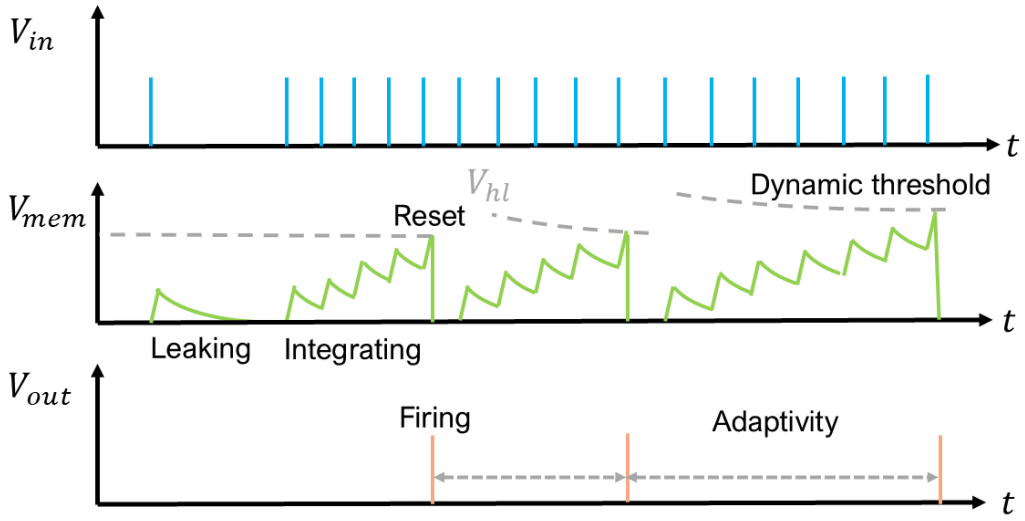

Figure 3: Idealized behavior of a LIF-Neuron when the input, V_{in}, is a spike train. V_{mem} is the membrane potential and shows a charge accumulation matching the input spikes and a leakage when V_{in} is zero. V_{out} is the spiking produced by the LIF-neuron. Furthermore, when a output spike is generated, the membrane potential resets and the threshold for a new output spike increases consequently fufilling the adaptation criteria

In addition to the above, there are three functional criteria on the interplay of neurons identified by teh project supervisor Libo Chen. They consist of:

- **Reception of excitatory and inhibitory spikes:** The accumulation of charge should be promoted/inhibited by inputs from other stimuli or neurons
- **Ability to drive subsequent neurons:** The neurons should be able to be connected in series and the output of multiple neuron spikes should trigger one spike in the subsequent neuron
- **Creation of rich spiking patterns:** The combination of the two previous function and neuron dynamics should create a output which can be temporally adjusted, thus creating a plethora of spiking patterns which can be used to carry information.

Excitatory spikes are the standard mode of operation and the only implemented spike type in this proposed circuit. To implement inhibitory spikes, one could connect the spikes to a RC circuit that is connected to the MOSFET gate threshold with a diode to prevent charging of the capacitance from the output voltage. With these additions one could create a circuit that is adaptable and interconnected with excitatory and inhibitory effects.

There exist two broad classes of neurons, called type I and II. They are distinguished by their relation between input current and the frequency of the output spikes. For both types, a certain input current is required to get spikes in the output. The difference is that for type I, the frequency increases continuously from zero above the minimal current, while type II jumps discontinuously to a high frequency at the minimal current [10].

Lastly, a more qualitative feature of biological neurons is that they are sparse: the width of the spike should be much smaller than the total frequency. This is an important factor in the energy efficiency of biological neurons. In this work, the aim is to replicate this feature, and so an ideal ratio between the spike width and the spike interval has been defined as 1:9. In subsequent discussions on frequency, we attempt to keep this ratio.

2.4 Advanced circuit

The advanced circuit differs from the basic circuit in its inclusion of a second hysteresis device. This replaces the MOSFET, and should give the circuit new properties. The new circuit diagram can be seen in Figure 4.

The magnitude of the current that discharges the capacitor is now determined by the second hysteresis device H_2. Mathematically, this is modeled by changing the expression of I_{DS} to be given by the output of this hysteresis device and not by a MOSFET equation:

$$I_{DS} = H_2(V_{out}, t), \tag{9}$$

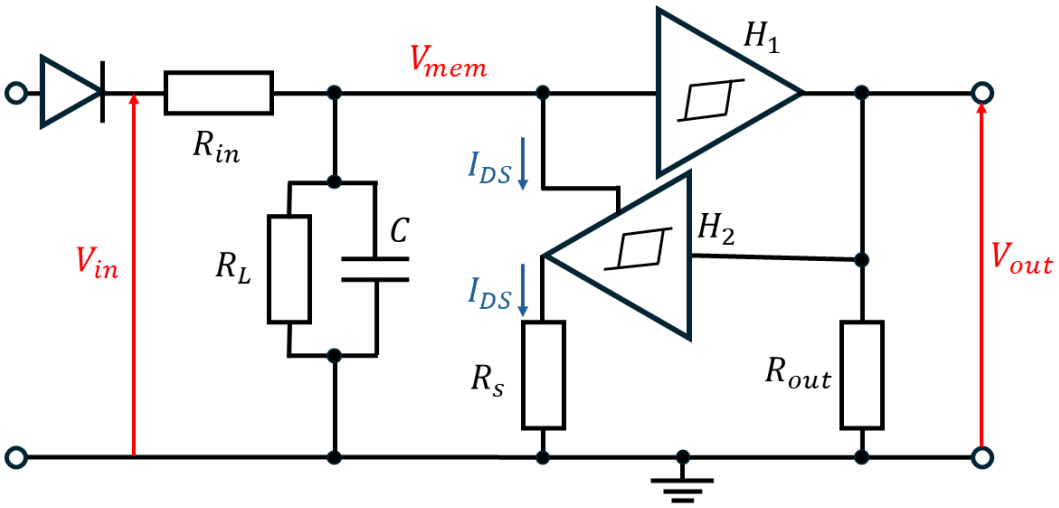

Figure 4: Sketch of the advanced LIF circuit. Variables are as in Figure 2, with the MOSFET replaced with a second hysteresis device. The old and new hysteresis devices are labeled H_1 and H_2 respectively.

where H_2 is the function for the I-V curve of the second hysteresis device. Denoting this current I_{DS} is perhaps no longer intuitive since drain and source (D and S) refer to the MOSFET, but we keep the name even if its meaning is lost.

2.5 State of the art – Loihi

Loihi is a neuromorphic chip that is built for spiking neural networks (SNN) [4]. The chip shows lower energy delay product by orders of magnitude compared to other models ran on CPU/GPU and is strongest in sparse event driven cases [4]. There are 3 common ways that the SNN is trained on the chip. These methods include: transforming an existing artificial neural network (ANN) trained on CPU/GPU, directly training a SNN offline, and finally training the SNN online, directly on the Loihi architecture with approximations of backpropagation.

When converting an existing ANN one has to note that if the ANN is a deep neural network (DNN) that has dense matrices, it will need multiple time steps to achieve the accuracy of the ANN and thus give a higher latency, even if the energy efficiency is generally higher then running the ANN [4]. Moreover, training the SNN offline takes advantage of the temporal encoding of the spikes and is optimal for sparse event activity. The cons of this method is that the training is costly and thus scaling to large networks is challenging. Finally, training the device online has advantages in robotics where the ”learning” can be done in real-time due to the programmable plasticity of the chip, thus approximating a backward propagation.

Where SNN models truly shine is when modeling phenomena that are naturally spiking and dynamic. These phenomena include attractor networks such as dynamic neural fields (DNFs) and locally competitive algorithms (LCAs) [4]. The attractor network leverages the dynamics of spiking neurons as well as their temporal aspects to solve tasks by converging to a steady solution in phase space. The LCA spreads an input across several neurons, thus creating a sparse mapping of the input which prevents overfitting and optimizes energy consumption and latency. This algorithm is optimal for usage with Loihi chips as the strength of the SNN is sparse event driven systems, this makes the energy efficiency and latency orders of magnitude better than running FISTA on CPUs [4]. DNF are attractor networks that use stable states, relations, behavior and memory. This can then be used to implement a single attractor such as object location or self-motion where the excitation of the input inhibits less stable states, thus ”focusing” on the relevant variable as in the winner-take-all (WTA) network operates.

3 Method

Building on the theory above, a Python script was written to simulate the time evolution of an artificial neuron circuit. This section describes the overall structure of the code, together with partial results showing the functionality of smaller code blocks, and is followed by a description of the workflow which produced the results.

3.1 Code

Here, the overall structure of the code is described, together with some examples of important functions used. The code was written from scratch in the language Python, and can be found in its entirety at GitHub[2].

To simulate the circuit and its components, a class-oriented approach was decided upon. This was because the whole circuit consists of many components with different components and internal states, that would be difficult to survey if everything was written inside one function. Three classes have been implemented: one representing the hysteresis device, one representing the MOSFET and one representing the whole LIF circuit (and thus including two instances of the prior classes).

3.1.1 Hysteresis device

The hysteresis device class was written first, according to the given specifications detailed in Section 2.1. To begin with, two hysteresis definitions were implemented in parallel: one where the vertical slopes are defined as input parameters and one where the two points, between which the vertical slopes are, are the input parameters. In the end, the former was decided upon because of its possibility to implement perfectly vertical slopes. An example I-V-curve is shown in Figure 5, with the shift after one loop shown. The voltage goes from 0 to 9 [a.u] and back thrice, with the results of the each loop shown in a different color. The parameters were chosen to give illustrative loops and were as such not necessarily ideal for use in a neuron circuit. The decay time τ was chosen much larger than the total simulated time t so that the decay of the shift should be negligible.

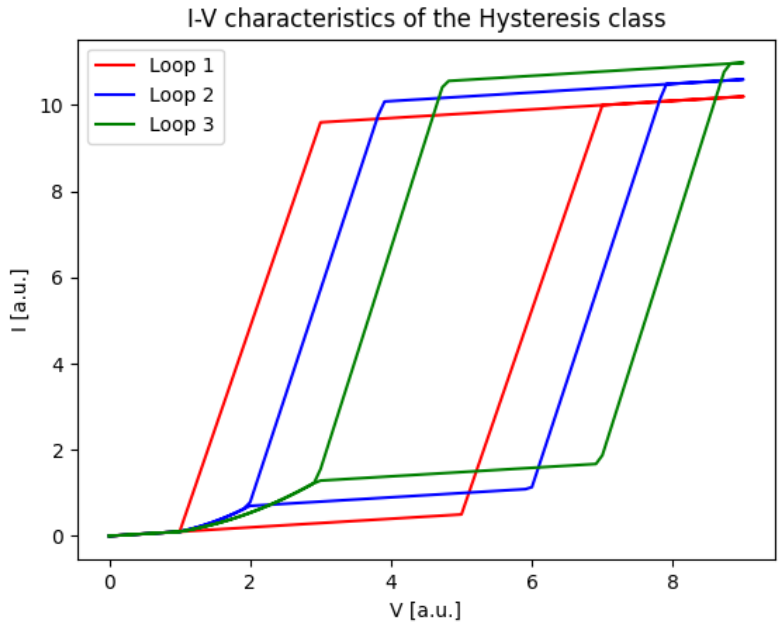

Figure 5: The I-V loop of the implemented hysteresis class with shift after each completed loop

Since the hysteresis loop gives different outputs for the same input depending on which path we are on, its internal state is stored in the boolean `stateOn`. If it is False, the output is taken from the lower path (from V_{start} to V_{sat} via V_{hl}). For an input voltage over V_{sat}, `stateOn` becomes True. Now, the output is taken from the upper path (from V_{sat} to V_{start} via V_{hl}). When the input voltage goes below V_{start}, `stateOn` becomes False again. At this point, one loop is considered finished and the time when it happened is recorded, for use in the shifting operation.

[2] `https://github.com/Hanedv01/Neuromorphic-Circuit-Design/tree/main`

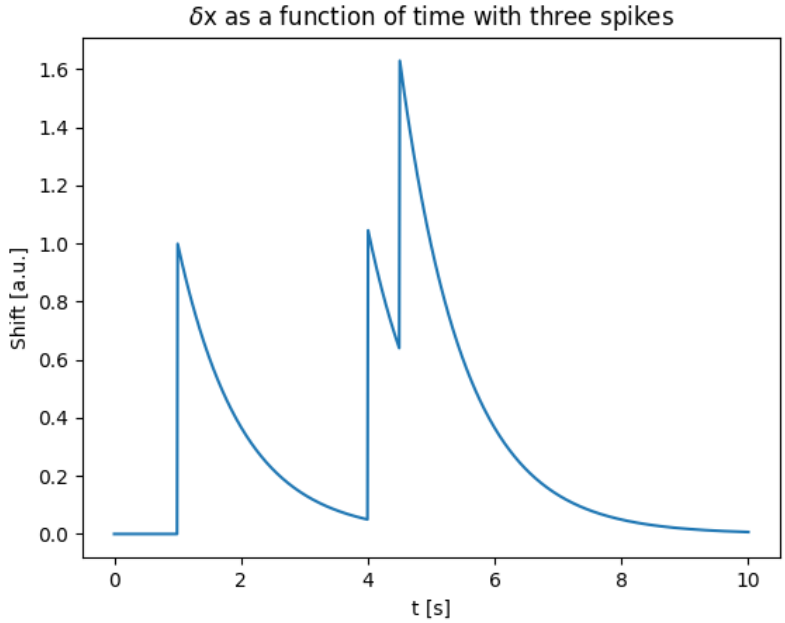

Figure 6: The shift function $\delta x(t)$ for spike times at $t = 1$, 4 and 4.5 s. The chosen parameter values are $\delta x_0 = 1$ and $\tau = 1$ s.

This shift is implemented as the method `Shift(t, delta, tau)`. `delta` is the variable name for the shift amplitude δx_0, and can be chosen to be either δI or one of the δV. An example plot showing the functionality can be found in Figure 6, where the spiking times have been chosen in advance. One can see that spikes in quick succession give a bigger net shift, but this still decays back to zero after some time.

3.1.2 MOSFET and LIF circuit

The MOSFET was implemented as a class, to bundle its parameters. It has a single method `GetIds(Vgs, Vds)` which returns the result of the Shichman-Hodges MOSFET equation (Equation 3). This method can raise two ValueErrors: one if V_{DS} is negative, which is not physical in this circuit, and one general ValueError if the inputs do not fall into one of the three cases in Equation 3. I_{DS} as a function of V_{DS} is shown in Figure 7.

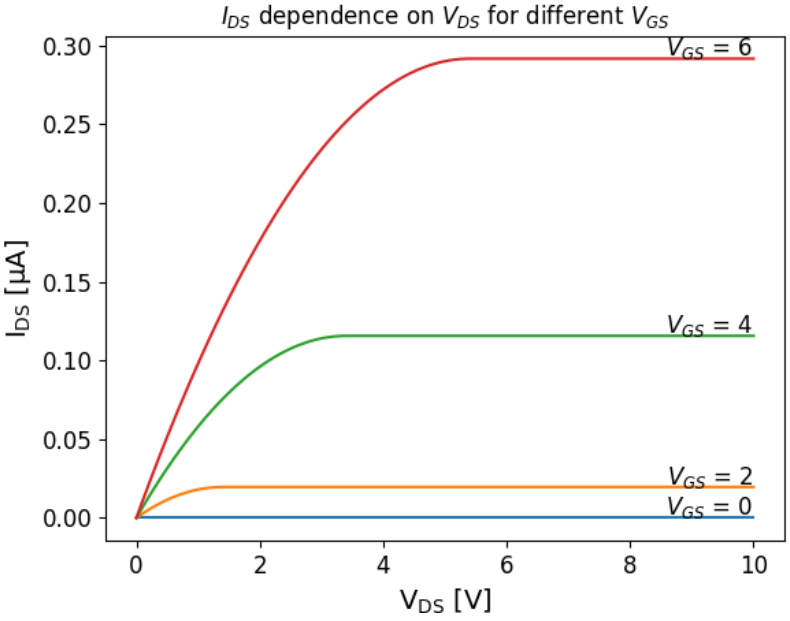

Figure 7: I_{DS} as a function of V_{DS} for different V_{GS} of the implemented MOSFET class. The chosen parameters were $V_{th} = 0.6$ V, $K = 2 \cdot 10^{-5} \frac{\text{A}}{\text{V}^2}$ and $\lambda = 0$.

The LIF circuit class consists of a hysteresis device, a MOSFET and the electronic components of the circuit, as seen in Figure 2. It has two important methods: `Step(t, dt, Vin)` updates the circuit for each time step according to Equation 2 and gives V_{mem} and V_{out} for the new time step. The differential equation is solved using Euler forward. As stated in the theory section, the dynamics of V_{mem} depend on I_{DS}, which recursively depends on itself. To solve for I_{DS}, the second method `SolveRecursiveIds(Vmem, Vout)` was implemented. It can be seen in Listing 1.

```
def SolveRecursiveIds(self, Vmem, Vout):
       def GetCircuitIds(Ids):
           Vgs = Vout - Ids*self.Rs
           Vds = Vmem - Ids*self.Rs
           if abs(Vds) <= 1e-15:   # Necessary for stability
               Vds = 0
           return self.MOSFET.GetIds(Vgs,Vds)

       def residual(Ids):
           return Ids - GetCircuitIds(Ids)

       Imax = Vmem/self.Rs
       sol = root_scalar(residual, bracket=[-Imax,Imax], method='brentq')

       if not sol.converged:
           raise RuntimeError("Ids solution did not converge.")
       return sol.root
```

Listing 1: The code solving for I_{DS}

Since I_{DS} in Equation 3 depends on itself as well as V_{mem} and V_{out}, a residual function `residual(Ids)` is defined as a subfunction which gives the difference between I_{DS} and the $I_{DS}(V_{mem}, V_{out}, I_{DS})$ which comes from Equation 3. This residual is minimized using a built-in function from `SciPy` called `root_scalar`. This function minimizes the residual, returning the I_{DS} where the residual is zero, using Brent's method [11]. Brent's method requires a bracket to search over and guarantees convergence if the function changes sign in over the interval. The bracket was chosen as $[-V_{mem}/R_S, V_{mem}/R_S]$, which is the maximum possible current, possible only when the internal resistance of the MOSFET is zero.

With functions that give I_{DS} and the dynamics of V_{mem}, the whole circuit is implemented. What remains is to systematically perform different simulations and to record the results.

3.2 Workflow

When the solver and equations where completed and the code was ready to run, the first task was to investigate when spiking behavior was achieved and what could cause this to fail. The results of this investigation can be found in Section 4.1.

When spikes in the output had been achieved, some reasonable initial parameters were needed to achieve a optimal spiking pattern. As stated previously, a desired ratio of 1:9 between spike width and spike interval was decided on. For simplicity and to operate at low voltages, it was decided that the input voltage would be 1 V. To achieve a 1 V voltage in the output spike, I_{sat} was chosen as 1 A and R_{out} was chosen equal to 1 Ω, making the numerical values of I_{out} in A equal to V_{out} in V. This gives a similar scale between input and output voltages while keeping calculations easy: 1 V output spikes can be achieved for any I_{sat} by adjusting R_{out}.

Furthermore, to maximize the frequency of the spikes a low capacitance of 10 pF was chosen. Finally, the MOSFET was chosen to have a gain of 2e-5 A/V^2 and a threshold voltage of 0.7 V, while the emerging device voltages where set to $V_{lh} = 0.1$ V and $V_{hl} = 0.7$ V. Then the leak resistance was set to 1e32 Ω to prevent any leaking and the other resistances were set to achieve the correct ratio of 9. The chosen initial parameters for the basic simulations can thus be seen in Section 4, Table 1.

After the decision regarding the initial parameters, the resistances, capacitance, MOSFET and hysteresis parameters where adjusted one by one to see the effects of these parameters on the spike width, spike interval and frequency. After deciding the effects of the parameters, some more advanced plots where made, deciding how the parameters affect each other and the operational area of the parameters depending on each other. Finally, graphs were made to determine if the neuron is a type I or type II neuron by looking at if the output frequency starts at zero or not.

4 Results and Discussion

In the following section, the results are presented together with a discussion on their implications. We begin by examining cases where desired effects could not be obtained, and the reasons why. Following that, the effects of the different parameters are discussed.

4.1 Obtaining correct outputs

The widest definition of a correct output is that the code should not give any errors. A slightly higher standard is that the circuit should exhibit spiking behavior for a constant input. This means that the hysteresis device should reach its on-state and from there be able to enter the off-state. The cases where this does not happen are discussed in this section.

4.1.1 Avoiding errors

The main error that the code can give is the `ValueError` implemented in the function for I_{DS} regarding V_{DS}. For our MOSFET model, V_{DS} is defined to be non-negative, so this error was included in the function as a notice that the arguments are not right. From Equation 5, V_{DS} is given as $V_{DS} = V_{mem} - R_D I_{DS}$. This means that two terms can cause in this error.

If V_{mem} is (too) negative, this error gets raised. This can be fixed by lowering $\mathrm{d}t$, as the dynamics of V_{mem} are given numerically by $\mathrm{d}V_{mem} = \ldots \mathrm{d}t$ and a smaller $\mathrm{d}t$ lowers the probability of overshooting during the discharging of the capacitor and getting a negative V_{mem}.

If on the other hand the term $R_D I_{DS}$ is too large, the same error is raised. The main culprit here is I_{DS}, since increasing R_D decreases I_{DS}. Having a too large K is usually what causes this, since I_{DS} scales linearly with K (see Equation 3).

4.1.2 Handling latch-up

Avoiding errors in the calculation is not a guarantee that we get the spiking output we desire. Even if the code can generate a solution, that solution can get stuck at a certain value for a constant input. This is commonly called latch-up in electronics.

One possibility is that V_{mem} never reaches V_{hl}, so the hysteresis device never turns on. An example of this is shown to the left in Figure 8. This only happens for very low R_L, which causes a large leak current, or very high V_{hl}. This case rarely came up in the actual examined cases, since R_L was usually kept high.

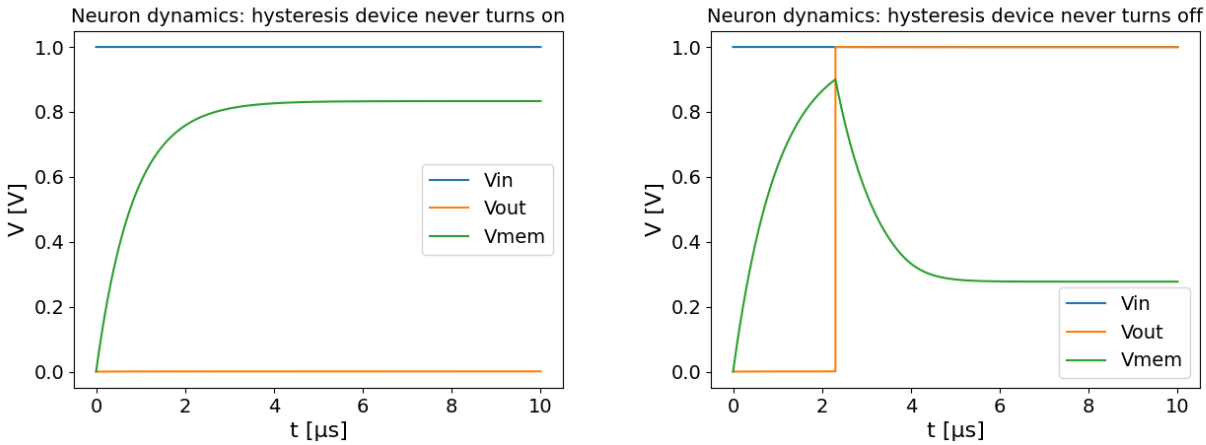

Figure 8: Two examples of neurons that get stuck for constant inputs. Left: V_{mem} never reaches V_{hl} (chosen as 0.9 V here), so the hysteresis device never reaches its on-state. Right: The hysteresis device reaches its on-state, but V_{mem} decays to a value higher than V_{lh} (chosen as 0.2 V here).

A more common issue was that the hysteresis device turned on and was stuck there. When the hysteresis device turns on, the capacitor discharges but V_{mem} approaches an asymptote above zero. If V_{lh} is chosen

below this asymptote, the device becomes stuck in the on-state. An example case is shown to the right in Figure 8. The value of this asymptote can be calculated with Equation A.1, as detailed and derived in Appendix A.2.

In Figure 9, the dependence of this asymptote on two key parameters is plotted. While these graphs are dependent on the input parameters, the trend is clear: if the neuron gets stuck in the on-state, one should attempt to increase R_{in} or K.

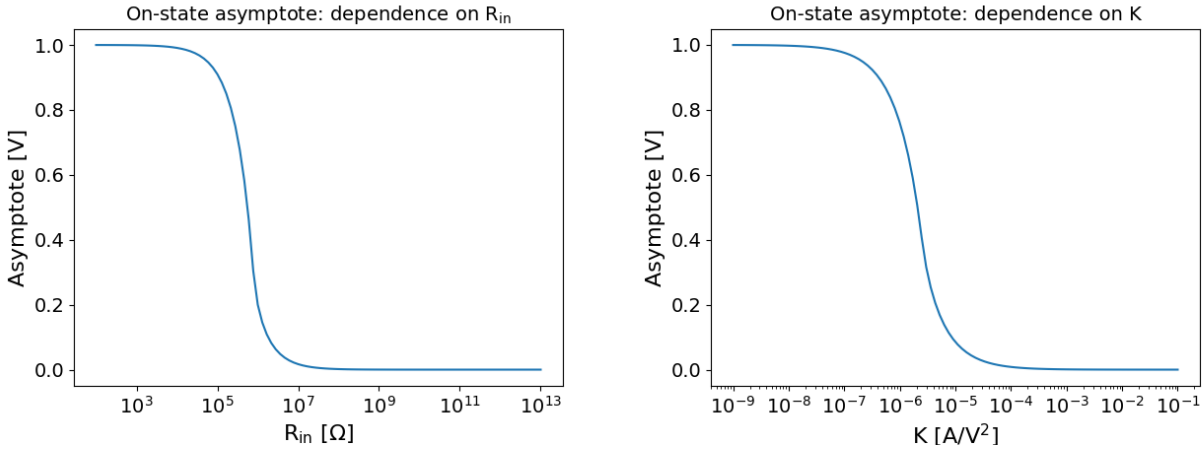

Figure 9: The dependence of the on-state asymptote on R_{in} (left) and K (right) respectively

4.2 Demonstration of spiking outputs

If the parameters of the hysteresis device are chosen correctly with regard to the asymptote locations, a spiking output can be obtained. Two example plots are given in Figure 10. The parameters were chosen to show both charging and discharging clearly and are not realistically sparse, in the previously defined meaning of having a ratio 1:9 between spike width and spike interval.

In both plots, the input voltage is constant and all parameters are the same, apart from the two δV. In the left plot, both δV are set to zero and the neuron thus shows no adaptability; the output frequency stays constant. V_{lh} and V_{hl} were set to 0.3 and 0.6 respectively, and both values are seen clearly in for which V_{mem} the spike turns off and on. In the right plot, the δV are non-zero, which leads to adaptability. The time between subsequent spikes is increased, as V_{lh} and V_{hl} increase and V_{mem} thus needs to charge further to reach the spiking threshold.

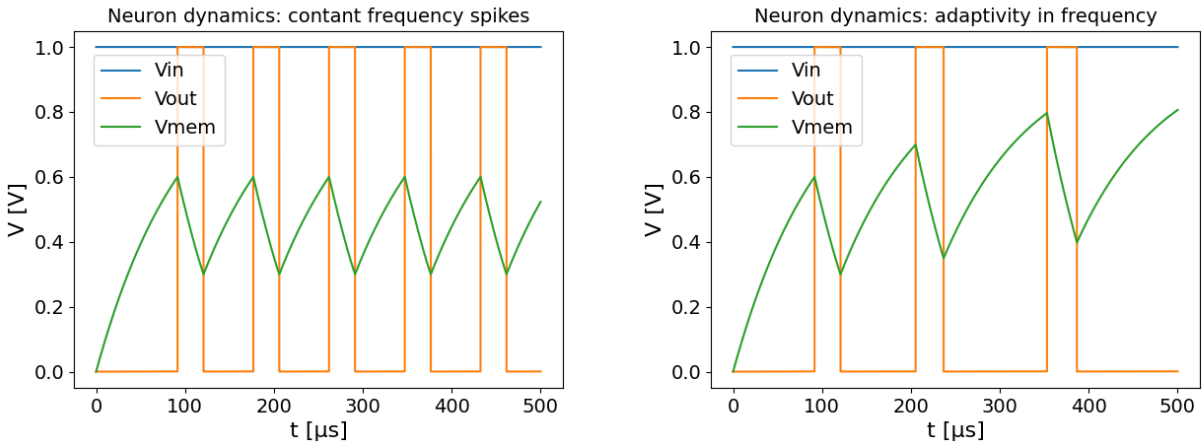

Figure 10: Demonstration of a spiking output for a constant input.

A final demonstration of the capabilities of this artificial neuron can be found in Figure 11. The similarities to the ideal behavior in Figure 3 should be obvious. One input spike charges the membrane, which then discharges exponentially. Multiple input spikes can cause a spike in the output, which changes the firing threshold of the

neuron. This can cause later output spikes to require more input spikes, leading to adaptation to continuous stimuli.

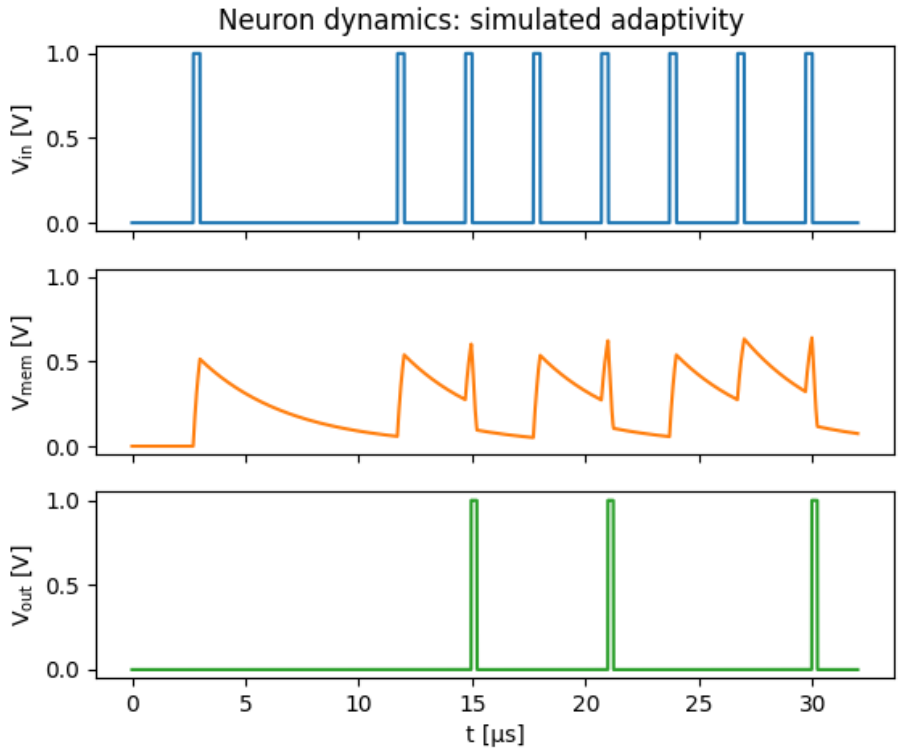

Figure 11: Demonstration of spiking output with adaptability for a spiking input with 250 kHz.

4.3 Effect of parameters

The parameters are dependent on each other but by changing one at the time, as in table 2 – 9 with initial parameters in table 1, one can compensate for parameters locked by components used in the circuit such as the available MOSFET transistors. Moreover, R_L has been set to 1e32 Ω to prevent leak current for these simulations. An example of parameter compensation is changing R_S and R_{in}, observed in Tables 2 and 3, which alters the spike width and time between peaks to a certain extent. Furthermore one can calculate the energy of the spike and the power consumption of the spike generation which are tightly correlated with the spike width and frequency as mentioned in section 2. Note that the spike energy is almost equal to the spike width since the output voltage is taken to be equal to the output current which is close to 1 at both V_{sat} and V_{lh}.

Moreover, it is important to note the time-step with which the simulations where performed. This is because possible errors could be in the order of the timestep size and the numerical stability of the Euler forward solver requires the time step to be sufficiently small to accurately simulate the behavior of the circuit.

Table 1: Initial parameters

Parameters	Initial value
R_{out}	1 Ω
R_S	100 Ω
R_{in}	7e6 Ω
R_L	1e32 Ω
K	2e-5
V_{th}	0.7 V
V_{lh}	0.1 V
V_{hl}	0.7 V
C_{mem}	1e-11 F
Spike width	7.96 μs
Spike interval	77.0 μs
Frequency	11.8 kHz
Spike energy	8.19 nJ
Power consumption	110 MW
Time-step size	39.1 [ns]

Table 2: Source resistance applied after the MOSFET, R_S

R_S [Ω]	Spike width [μs]	Spike interval [μs]	Spike energy [nJ]	Power consumption [μW]	Time-step size [ns]
0.1	8.01	77.0	7.99	94.1	39.1
1	8.01	77.0	7.99	94.1	39.1
10	8.01	77.0	7.99	94.1	39.1
100	8.01	77.0	7.99	94.1	39.1
1e3	8.07	77.0	8.07	94.8	39.1
1e4	8.56	77.0	8.54	99.9	39.1
1e5	13.4	77.0	13.4	148	39.1
1e6	Constant peak	-	-	-	-

Table 3: Resistance before entering the circuit, R_{in}

R_{in} [Ω]	Spike width [μs]	Spike interval [μs]	Spike energy [nJ]	Power consumption [μW]	Time-step size [ns]
1e6	constant peak	-	-	-	-
7e6	8.01	77.0	7.99	94.1	39.1
1.5e7	7.50	165	7.49	43.4	39.1
5e7	7.23	550	7.21	12.95	39.1
1e8	7.19	1100	7.17	6.48	39.1

Furthermore, changing the MOSFET gain, K, and threshold voltage, V_{th}, seen in Table 6 and 7 in Appendix B, affects the width peak without changing the distance between peaks. Another observation can be inferred from Tables 8 and 9 in Appendix B, showing the adjustments of V_{hl} and V_{lh}. The observation is that a small difference between V_{hl} and V_{lh} naturally gives a tighter peak but also changes the spike interval due to the charge difference between on and off states being smaller so charging also speeds up. One can also infer that the difference between I_{on}(V) and I_{off}(V), i.e. the height of the hysteresis curve, provides the amplitude of the peak.

In addition, V_{mem} discharges exponentially, which makes it more optimal to increase V_{lh} than to decrease V_{hl} to obtain a high frequency (i.e. short peaks and cooldown). Consequently, if a high frequency is desired one should manufacture the hysteresis device as small and tall as possible. Then after the device is manufactured one has to tune R_{in} to change the time between peaks.

It would also be optimal to increase the MOSFET gain, K, and lower the MOSFET threshold, V_{th}. However, one should confirm the intended values with simulations before manufacturing as R_{in} can only be lowered to a

certain extent before no peaks are observed, which is dependent on other parameters in the circuit. Moreover, lowering C_{mem} will also increase the frequency of the peaks. However, lowering C_{mem} further than 10 pF could be problematic since the circuit introduces parasitic capacitance which is not taken into account for in the simulations.

To substantiate the importance of the MOSFET parameters, a simulation was completed lowering the MOSFET gain, K and the threshold voltage, V_{th}, which can be seen in table 4. To further concretize the point made about V_{hl} and V_{lh}, two more simulations with constant voltage gap between these two parameters and the same changed MOSFET parameters where made and the results can be observed in table 5. From this we see that increasing V_{lh} can substantially increase the frequency compared to decreasing V_{hl}, if one keeps the same ratio between spike interval and spike width to be around 9 by adjusting R_{in}.

Table 4: Adjustments to the MOSFET parameters and R_{in}

K	2.5e-4
V_{th}	0.5 V
R_{in}	2.3e5 Ω
Spike width	284 ns
Spike interval	253e1 ns
Frequency	0.356 MHz
Spike energy	283 pJ
Power consumption	101 μW
Time-step size	61.0 ps

Table 5: Adjustments of V_{hl} and V_{lh} parameters while keeping the modified MOSFET parameters from table 4 except R_{in}.

(a) Decreasing V_{hl}

R_{in}	4e5 Ω
V_{hl}	0.35 V
Spike width	144 ns
Spike interval	1300 ns
Frequency	0.692 MHz
Spike energy	144 pJ
Power consumption	99.7 μW
Time-step size	61.0 ps

(b) Increasing V_{lh} while preserving the same V_{hl}–V_{lh} difference as in Table 5a.

R_{in}	1.35e5 Ω
V_{lh}	0.45 V
Spike width	90.3 ns
Spike interval	819 ns
Frequency	1.10 MHz
Spike energy	90.3 pJ
Power consumption	99.3 μW
Time-step size	61.0 ps

4.4 Determination of neuron type

As stated in the theory section, there exist two types of neuron. To determine which of them this model represents, the dependence of the output frequency on the amplitude of the constant input voltage was measured, the results of which can be found in Figure 12. It is clear that this model represents a type II neuron, with the lowest frequency clearly larger than zero.

4.5 Advanced circuit behavior

The advanced circuit can produce spiking outputs, just like the basic one. To only focus on the dynamic effect of the second hysteresis device, the first hysteresis device was chosen to have no shifts after each spike (for exact values, see Appendix C). An example spiking pattern can be seen in Figure 13.

From the figure, it is clear that only having δI in the second hysteresis device makes each subsequent spike shorter. In Figure 13, the first spike is 67 ns long while the ninth and last is 30 ns. In contrast, the time between spikes is constant. This means that a second hysteresis device allows us to control the adaptivity of the spike width independently from other parameters, which could allow for more advanced spiking patterns and learning in a neural network.

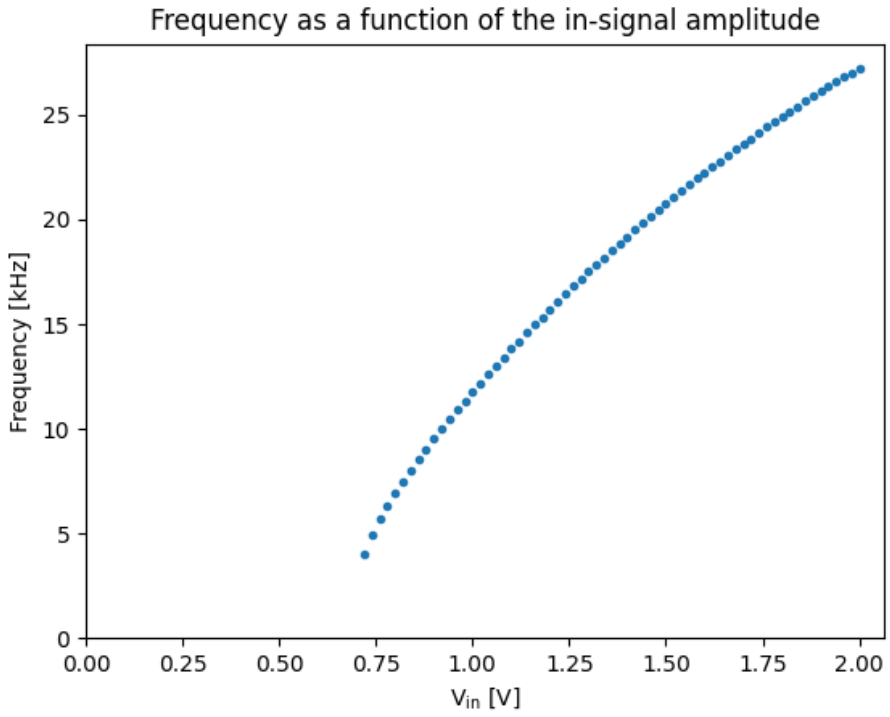

Figure 12: The dependence of output spike frequency as a function of constant input voltage

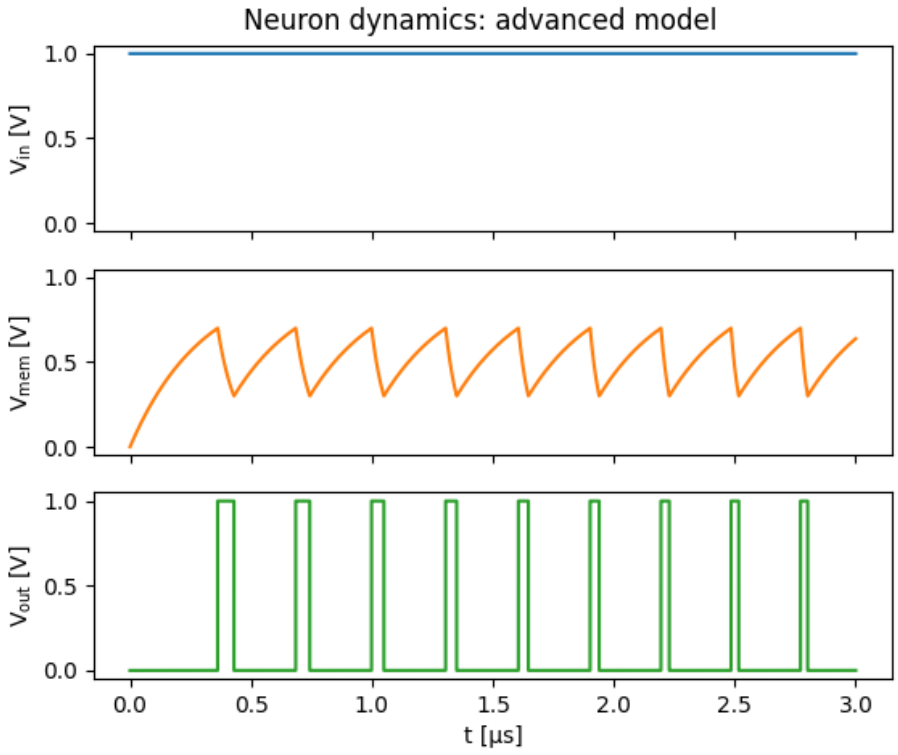

Figure 13: The spiking pattern of the advanced model with adaptability in δI

5 Conclusion

In this project we explained the need for neuromorphic circuit design and explained how this could be realized using an emerging device. Furthermore, a description of a circuit for a LIF-neuron was made and a simulated with the mentioned characteristics. The simulations showed that the LIF-neural circuit acted as intended, leaking, integrating and firing. Moreover, the neuron adaptation and threshold dynamics together with the ability to drive subsequent neurons were shown for the circuit. Other observations from the simulations gave an insight into how the different parameters affect the spiking pattern and frequency. There was also insight into how to balance the ratio between spike width and spike interval, and how to increase the frequency of the spikes. Replacing the MOSFET with a second hysteresis device allows one to control the adaptability in spike width as well.

Note however that the model used for the MOSFET and emerging device where simplified, and thus the values of the parameters for when the circuit works might be less flexible in experiments. It is also important to take the time-step length into consideration for the accuracy of the results and for the convergence of the Euler forward model used in the simulation.

The theoretical results in the form of spiking patterns are similar to what has been achieved experimentally in other studies using other artificial neurons, such as [6]. The next step could thus be to design a proof of concept experiment using readily available components (such as Schmitt triggers) and compare the experimental results with that of the simulation to determine the accuracy of the simple model used for the MOSFET transistor and emerging device. If that succeeds, the next logical step is to create a physical prototype using a more compact and energy efficient hysteresis device, as a first step in creating a new, more efficient hardware for artificial intelligence.

References

[1] Noble Prize Outreach. Nobel prize in physics 2024 [internet]. Available from: https://www.nobelprize.org/prizes/physics/2024/summary/, 2024. [Cited 2025 November 17].

[2] IEA. Energy and AI. Available from: https://www.iea.org/reports/energy-and-ai, 2025. [Cited 2025 November 17].

[3] Michael V. DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P. Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K. Nayak, Rathinakumar Appuswamy, Peter J. Carlson, Andrew S. Cassidy, Pallab Datta, Steven K. Esser, Guillaume J. Garreau, Kevin L. Holland, Scott Lekuch, Michael Mastro, Jeff McKinstry, Carmelo di Nolfo, Brent Paulovicks, Jun Sawada, Kai Schleupen, Benjamin G. Shaw, Jennifer L. Klamo, Myron D. Flickner, John V. Arthur, and Dharmendra S. Modha. Truenorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5):20–29, 2019. doi: 10.1109/MC.2019.2903009.

[4] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021. doi: 10.1109/JPROC.2021.3067593.

[5] Yixin Zhu, Huiwu Mao, Ying Zhu, Xiangjing Wang, Chuanyu Fu, Shuo Ke, Changjin Wan, and Qing Wan. Cmos-compatible neuromorphic devices for neuromorphic perception and computing: a review. *International Journal of Extreme Manufacturing*, 5(4):42010–, 2023. ISSN 2631-8644.

[6] Rongrong Cao, Xumeng Zhang, Sen Liu, Jikai Lu, Yongzhou Wang, Hao Jiang, Yang Yang, Yize Sun, Wei Wei, Jianlu Wang, Hui Xu, Qingjiang Li, and Qi Liu. Compact artificial neuron based on anti-ferroelectric transistor. *Nature communications*, 13(1):7018–10, 2022.

[7] H Shichman and DA Hodges. Modeling and simulation of insulated-gate field-effect transistor switching circuits. *IEEE journal of solid-state circuits*, (3):285–289, 1968.

[8] Bogdan M. Wilamowski. Models of semiconductor devices [lecture notes]. Available from: https://www.eng.auburn.edu/~wilambm/pap/1997/SBook/6_SEM.PDF, 1997. [Cited 2025 November 19].

[9] Ruoyu Zhao, Tong Wang, Taehwan Moon, Yichun Xu, Jian Zhao, Piyush Sud, Seung Ju Kim, Han-Ting Liao, Ye Zhuo, Rivu Midya, Shiva Asapu, Dawei Gao, Zixuan Rong, Qinru Qiu, Cynthia Bowers, Krishnamurthy Mahalingam, S. Ganguli, A. K. Roy, Qing Wu, Jin-Woo Han, R. Stanley Williams, Yong Chen, and J. Joshua Yang. A spiking artificial neuron based on one diffusive memristor, one transistor and one resistor. *Nature Electronics*, 2025.

[10] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal Dynamics*. Cambridge University Press, 2014. 4.4 Type I and Type II neuron models.

[11] The SciPy community. root_scalar (scipy v1.16.2 manual). Available from: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root_scalar.html, n.d. [Cited 2025 December 2].

[12] MathWorks, Inc. N-channel mosfet. Available from: https://se.mathworks.com/help/sps/ref/nchannelmosfet.html, 2025. [Cited 2025 November 25].

A Appendix: Mathematics

In this appendix, the mathematical details of the theory section are given in more detail.

A.1 Derivation of the LIF circuit equations

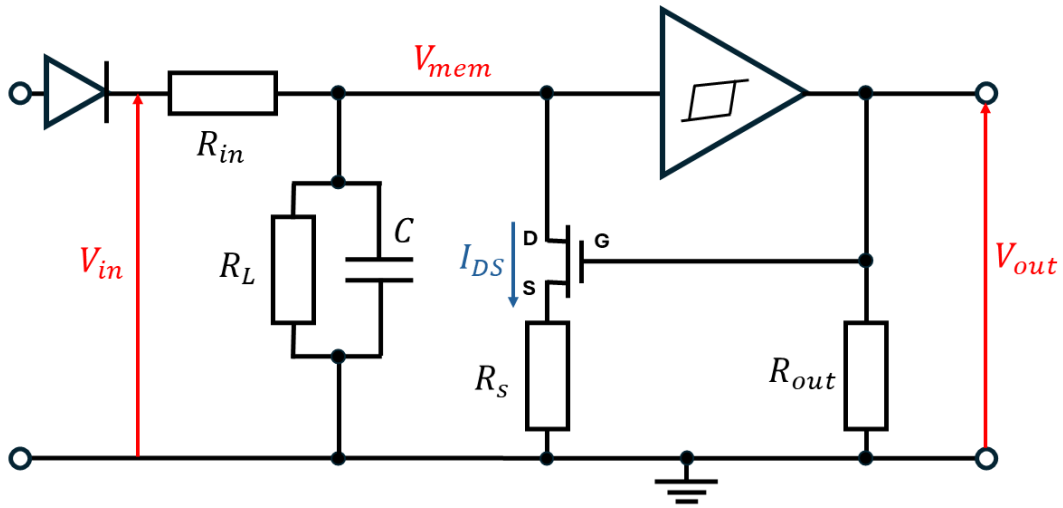

Figure 2: A sketch of the LIF circuit (repeated)

A sketch of the LIF circuit was given in Figure 2, repeated here for convenience. Kirchhoff's current law for the junction V_{mem} gives

$$I_{in} = I_{R_L} + I_C + I_{DS},$$

where I_{in} is the current through R_{in}, I_{R_L} and I_C are the currents through the components with the same names and I_{DS} is the drain-source current through the MOSFET, marked in the sketch. I_{in} and I_{R_L} are given by Ohm's law, while I_C is the current to a capacitor:

$$I_{in} = \frac{V_{in} - V_{mem}}{R_{in}} \qquad I_{R_L} = \frac{V_{mem}}{R_L} \qquad I_C = C\frac{\mathrm{d}V_{mem}}{\mathrm{d}t}.$$

Combining these and rearranging, we get

$$\frac{\mathrm{d}V_{mem}}{\mathrm{d}t} = \frac{V_{in} - V_{mem}}{CR_{in}} - \frac{V_{mem}}{CR_L} - \frac{I_{DS}}{C}.$$

This is essentially Equation (2). To model the effect of the diode, which does not let the capacitor discharge through R_{in}, an additional clause is added: if $V_{mem} \leq V_{in}$, meaning that the current would go from V_{mem} to V_{in}, we set $V_{in} = V_{mem}$. Effectively, this removes the first term and disallows capacitor discharging through R_{in}. The total equation thus becomes

$$\frac{\mathrm{d}V_{mem}}{\mathrm{d}t} = \begin{cases} \dfrac{V_{in} - V_{mem}}{R_{in}C} - \dfrac{I_{DS}}{C} - \dfrac{V_{mem}}{R_L C}, & V_{in} > V_{mem} \\ -\dfrac{I_{DS}}{C} - \dfrac{V_{mem}}{R_L C}, & V_{in} \leq V_{mem} \end{cases}. \tag{1}$$

Thus far, the only undetermined quantity in this equation is I_{DS}. There are many models for FET devices; we have opted for the Shichman-Hodges model of a N-channel MOSFET for its simplicity [12]. The equations giving I_{DS} for a positive V_{DS} are below, taken directly from the model:

$$I_{DS} = \begin{cases} 0, & V_{GS} < V_{th} \\ K\Big((V_{GS} - V_{th})V_{DS} - V_{DS}^2/2\Big)(1 + \lambda V_{DS}), & 0 < V_{DS} < V_{GS} - V_{th} \\ (K/2)(V_{GS} - V_{th})^2(1 + \lambda V_{DS}), & 0 < V_{GS} - V_{th} < V_{DS} \end{cases} \tag{2}$$

We set $\lambda = 0$ in our calculations. By definition, $V_{GS} \equiv V_G - V_S$ and $V_{DS} \equiv V_D - V_S$. The voltage over R_S is given when I_{DS} is known: $V_S = R_S \cdot I_{DS}$. Thus,

$$V_{GS} = V_{out} - R_D I_{DS} \tag{3}$$
$$V_{DS} = V_{mem} - R_D I_{DS}. \tag{4}$$

Lastly, the output voltage V_{out} is given by the hysteresis function we define, with V_{mem} as input:

$$V_{out} = R_{out} \cdot H(V_{mem}). \tag{5}$$

Since we can scale the output current of the hysteresis device arbitrarily and we can choose R_{out} freely, we use the output of $H(V_{mem})$ directly as V_{out} in the code.

A.2 Derivation of the on-state asymptote

As stated in the results section, the circuit can get stuck in the on-state of the hysteresis device for a constant input. This happens when V_{mem} approaches a constant value that is larger than V_{lh} of the hysteresis device. Recall the first case of Equation 2:

$$\frac{\mathrm{d}V_{mem}}{\mathrm{d}t} = \frac{V_{in} - V_{mem}}{R_{in}C} - \frac{I_{DS}}{C} - \frac{V_{mem}}{R_L C}.$$

Since this asymptote only exists for a constant input V_{in}, we do not need to consider the case $V_{mem} > V_{in}$ since that cannot happen for a constant input. For the asymptote, $\frac{\mathrm{d}V_{mem}}{\mathrm{d}t} = 0$, and denote the constant $V_{in} \equiv V_{in}^0$. Rearranging the above equation, we get

$$V_{mem} = \frac{\frac{V_{in}^0}{R_{in}} - I_{DS}}{\frac{1}{R_{in}} + \frac{1}{R_L}}. \tag{A.1}$$

Note that I_{DS} depends on both V_{mem} and V_{out} (in addition to itself). We can simplify this by assuming that V_{out} takes the constant value $R_{out} I_{sat}$. This requires a relatively large R_{on} so that the current out of the hysteresis device keeps close to I_{sat}, but since the asymptotic value is not used in further calculations, a small deviation can be tolerated. This means that I_{DS} effectively only depends on V_{mem} and itself. This recursive dependence can be solved by SciPy's function `root_scalar`, which uses Brent's method. Also of note is that the capacitance C does not appear in this expression, as it is present in all non-zero terms and can thus be canceled.

B Appendix: Results

In this appendix, some numerical results are recorded.

B.1 MOSFET and device Parameter adjustment tables

This section shows the spike width and spike interval for different values of MOSFET parameters and hysteresis device parameter

B.1.1 MOSFET

Table 6: Adjustments of the MOSFET gain, K

K [A/V^2]	Spike width [μs]	Spike interval [μs]	Spike energy [nJ]	Power consumption [μW]	Time-step size [ns]
1e-5	18.7	76.9	18.7	195	4.88
2e-5	8.06	76.9	8.04	94.6	4.88
4e-5	3.78	76.9	3.78	46.8	4.88
6e-5	2.47	76.9	2.47	31.1	4.88
2.5e-4	0.576	77.0	0.575	7.41	4.88
5e-4	0.288	77.2	0.288	3.71	4.88

Table 7: Adjustments of the MOSFET threshold voltage, V_{th}

V_{th} [V]	Spike width [ms]	Spike interval [ms]	Spike energy [nJ]	Power consumption [μW]	Time-step size [ns]
0.4	2.38	76.9	2.38	30.0	4.88
0.5	3.17	76.9	3.17	39.6	4.88
0.6	4.66	76.9	4.65	57.0	4.88
0.7	8.06	76.9	8.04	94.6	4.88
0.8	19.9	76.9	19.8	205	4.88
0.9	Constant peak	-	-	-	4.88

B.1.2 Hysteresis device

Table 8: Adjustments to the hysteresis device's on to off voltage, V_{lh}

V_{lh} [V]	Spike width [μs]	Spike interval [μs]	Spike energy [nJ]	Power consumption [μW]	Time-step size [ns]
0	Constant peak	-	-	-	4.88
0.1	8.06	76.9	8.04	94.6	4.88
0.2	6.16	68.7	6.15	82.2	4.88
0.3	4.84	59.3	4.83	75.4	4.88
0.4	3.59	48.5	3.59	68.9	4.88
0.5	2.37	35.8	2.37	62.2	4.88
0.6	1.18	20.2	1.18	55.0	4.88

Table 9: Adjustments of the hysteresis device's off to on voltage, V_{hl}

V_{hl} [V]	Spike width [μs]	Spike interval [μs]	Spike energy [nJ]	Power consumption [μW]	Time-step size [ns]
0.2	1.89	8.26	1.89	186	4.88
0.3	3.21	17.6	3.21	154	4.88
0.4	4.45	28.4	4.45	135	4.88
0.5	5.67	41.2	5.57	121	4.88
0.6	6.88	56.8	6.86	108	4.88
0.7	8.06	76.9	8.04	94.6	4.88
0.8	9.22	105	9.20	80.3	4.88
0.9	10.4	154	10.3	63.0	4.88
1.0	-	No spikes	-	-	4.88

C Appendix: Used parameter values

In this appendix, the parameter values used to obtain the figures in this report are found in Table 10, below. Unless specified, $V_{hl} = V_{sat}$ and $V_{lh} = V_{start}$. In Figure 5, this does not hold: the chosen values are $V_{start} = 1$ V and $V_{sat} = 7$ V. In all figures except Figure 12, the input voltage has an amplitude of 1 V. "**VAR**" means that that parameter varies in the plot, and L and R mean left and right subfigure respectively.

Table 10: Parameter values used to produce plots

Fig.	R_{in}	R_L	R_S	C	V_{hl}	V_{lh}	I_{sat}	R_{off}	R_{on}	τ	δV_{on}	δV_{off}	δI	V_{th}	K
5 *	–	–	–	–	5	2	10	10	10	1e3	1	1	0.5	–	–
8 L	1e6	5e6	1e1	1e-12	0.9	0.3	1	1e3	1e3	1e-1	0	0	0	0.6	2e-6
8 R	1e6	1e12	1e1	1e-12	0.9	0.2	1	1e3	1e3	1e-1	0	0	0	0.6	2e-6
9 L	**VAR**	1e12	1e3	1e-12	0.7	0.1	1	1e3	1e3	1e-1	0	0	0	0.7	2e-5
9 R	3e6	1e12	1e3	1e-12	0.7	0.1	1	1e3	1e3	1e-1	0	0	0	0.7	**VAR**
10 L	1e7	1e12	1e1	1e-11	0.6	0.3	1	1e3	1e3	1e-2	0	0	0	0.6	2e-6
10 R	1e7	1e12	1e1	1e-11	0.6	0.3	1	1e3	1e3	1e-2	0.1	0.05	0	0.6	2e-6
11	4e5	4e6	1e1	1e-12	0.6	0.1	1	1e3	1e3	1e-2	0.02	0.01	0	0.6	3e-5
12	7e6	1e32	1e2	1e-11	0.7	0.1	1	1e3	1e3	1e-2	0	0	0	0.7	2e-5