SYDE 556/750
Simulating Neurobiological Systems
Lecture 2: Neurons

Andreas Stöckel

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

January 9, 2020

UNIVERSITY OF
WATERLOO

Accompanying Readings: Chapter 4.1 of Neural Engineering

Contents

1   Introduction

Note: In the last lecture, we took a deep dive into neuroscience, and we saw how "messy" and complex nervous systems are. In this lecture we try to back up a little, moving us back to the safe haven of mathematical abstraction.

Neurons are the fundamental unit of computation in the nervous system. They compute by receiving action potentials (spikes) from pre-synaptic neurons in their dendrites. In most neurons, once the number of spikes received within a certain timespan surpasses a threshold, the neuron itself may emit an action potential that is being propagated along the neuron's axon to the axon terminal, where it will pass across a synapse to the post-neuron's dendrites where the process repeats (cf. fig. 1).

From an engineering perspective, we could say that individual neurons exchange coded information. An important part of understanding nervous systems is thus to understand the "code" that is being used for neural communication.

Unfortunately, there is no scientific consensus as for what exactly this "neural code" is. Most evidence points at a combination of population coding (i.e., information is encoded in the relative activities of a group of neurons) and time coding (i.e., the timing of individual spikes matters) [**rieke1999spikes**].

Note: There isn't a single neural code; different coding strategies are employed in different parts of the nervous system. Codes differ significantly between the peripheral nervous system (i.e., the sensory and motor neurons distributed throughout the body; especially the latter are clearly using a rate code) and the central nervous system.

What we do have however, are detailed models that describe how individual neurons generate action potentials. Thus, we will approach the problem of deciphering the neural code in two stages. First, in this lecture, we will have a look at single neurons and try to get an understanding of how neurons generate action potentials. We discuss the so called "Leaky Integrate-and-Fire" neuron, and summarize its behaviour in a simple analytical expression. Second, in the next two lectures, we think about the neural code in terms of *neural representation*, which will lead us to a theory that predicts what the neural code may be.

2   Spiking Neurons

Note: We're going to have a slightly closer look at biologically detailed spiking neuron models towards the end of the term. For now, we're skimming over the details. Have a look at [**kandel2012principles**] (particularly Chapter 7 and 8) if you want to learn more about basic neurobiology.

Neurons are cells that specialise in the integration and transmission of electrical signals. Cells in general are separated from the environment by a thick, impermeable "barrier", the *cell*

membrane, consisting of a bi-layer of lipid molecules. The cell membrane establishes an "intracellular" space that is isolated from the "extracellular" space. Both spaces are filled with a watery liquid, called the *intracellular fluid* and *extracellular fluid*, respectively (fig. 1).

2.1   Qualitative neural behaviour

When we insert an electrode into a resting neuron (akin to the "single electrode recording" we saw earlier), we can measure a difference in electrical potential, i.e., a voltage v, between the intra- and extracellular space (fig. 1). We call this voltage the *resting potential E_L*.[1]

Instead of just measuring this potential, we may also inject an external current into the neuron by hooking it up to a current source (i.e., a precision power supply that controls current instead of voltage). This is similar to what happens whenever a neuron receives a spike from a pre-synaptic neuron: the synapse induces either a positive ("excitatory synapse"), or a negative ("inhibitory synapse") current that flows into the neuron.

When injecting positive ("excitatory") currents into a neuron, we observe four major things:

1. The cell acts like a *capacitor*, i.e., the voltage increases while we're injecting a current (fig. 2a).

2. The capacitor is *leaky*. As soon as we stop injecting a current, the voltage collapses back to the resting potential E_L (fig. 2a).

3. As soon as the voltage surpasses a certain value, the *threshold potential v_{th}*, the cell will generate a spike (fig. 2b).

4. Shortly after the spike has been produced, the voltage drops below the resting potential. During this period, the *refractory period* of length τ_{ref}, we cannot get the neuron to spike again, even if we apply relatively large input currents J (fig. 2b).

---

[1] The symbol "E_L" stems from the alternative name of this potential, the "**L**eak channel **E**quilibrium potential".

> **Note:** Importantly, neurons in biology are *dynamical systems*, i.e., they possess a behaviour that evolves over time. *Artificial neurons* in machine learning (see below) are time-independent. They are mathematical functions that take an input that is "immediately" being mapped onto an output.

2.2 The Leaky Integrate and Fire Neuron

We can qualitatively summarize this behaviour in a very simple model, the so called "Leaky Integrate and Fire" neuron model. This model was first proposed by the French scientist Louis Lapicque in 1907 [**lapicque1907recherches**, **abbott1999lapicque**].

Sub-threshold behaviour   First, we have the *sub-threshold* behaviour, a so called leaky integrator.

$$\frac{\mathrm{d}}{\mathrm{d}t}v(t) = \frac{1}{C_\mathrm{m}}\big(g_\mathrm{L}(E_\mathrm{L} - v(t)) + J\big), \quad \text{if } v(t) < v_\mathrm{th}. \tag{1}$$

This differential equation corresponds to a capacitor with capacity C_m that is charged with a current J and that slowly discharges to a potential v_reset over a resistor with conductance (the inverse of the resistance) $g_\mathrm{L} = \frac{1}{R}$ (the *leak conductance*):
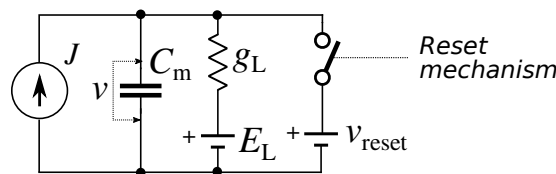
Super-threshold behaviour   Second, we have the *super-threshold behaviour*, i.e. the spike production and refractory period. Assume $v(t) = v_\mathrm{th}$ at $t = t_\mathrm{th}$. Then

$$\begin{aligned}
v(t) &\leftarrow \delta(t - t_\mathrm{th}), && \text{if } t = t_\mathrm{th}, \\
v(t) &\leftarrow v_\mathrm{reset}, && \text{if } t > t_\mathrm{th} \text{ and } t \leq t_\mathrm{th} + \tau_\mathrm{ref},
\end{aligned} \tag{2}$$

> **Note:** $\delta(t)$ is the Dirac delta function, i.e., the function defined as
>
> $$\delta(t) = \begin{cases} \infty & \text{if } t = 0 \\ 0 & \text{if } t \neq 0, \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(t)\,\mathrm{d}t = 1.$$

Normalized equations   For our modelling purposes, we don't really care about the exact values of v_th, v_reset and E_L. We can normalise these voltages, i.e., assume that $v_\mathrm{th} = 1$, and

$v_{\text{reset}} = E_{\text{L}} = 0$. Now we can simplify eqs. (1) and (2) to

$$
\begin{aligned}
\frac{\text{d}}{\text{d}t} v(t) &= -\frac{1}{\tau_{\text{RC}}}\big(v(t) - RJ)\big), && \text{if } v(t) < v_{\text{th}}, \\
v(t) &\leftarrow \delta(t - t_{\text{th}}), && \text{if } t = t_{\text{th}}, \\
v(t) &\leftarrow 0, && \text{if } t > t_{\text{th}} \text{ and } t \leq t_{\text{th}} + \tau_{\text{ref}},
\end{aligned}
\tag{3}
$$

where, $\tau_{\text{RC}} = C_{\text{m}}R$ and $R = \frac{1}{g_{\text{L}}}$.

2.3   Characterizing the firing-rate of a LIF Neuron

Figure 3 depicts the behaviour of the LIF neuron model eq. (3), as well as the detailed Hodgkin-Huxley type neuron model from before, as we slowly increase the input current J. Overall, we can see that the behaviour of the LIF neuron matches the behaviour of the more complex model quite well – at least qualitatively speaking.

Furthermore, we observe that the neurons fire quite regularly, yet the rate at which each neuron emits spikes (is "firing") increases as we increase the input current.

This raises the question: instead of dealing with these complicated differential equations, could we just summarize the behaviour of the neuron, e.g., compute the *firing rate a* for a given input current J? At least for the LIF neuron we can derive an analytic expression $G[J]$, the so called *neural response curve* that maps a current J onto the average spikes per second.

Note: Of course, with this *rate approximation*, we loose any information about spike timing. And having access to spike timing is still very useful when we want to bridge to lower levels of organization. We will deal with this in two weeks, when we talk about *temporal representation*. Furthermore, spiking communication induces *noise* that may be important from a modelling perspective.

Since the sub-threshold portion of eq. (3) is a simple first-order linear differential equation we can actually fully characterize the behaviour of the LIF neuron under the assumption of a constant input current. Particularly, we can derive a closed form expression that describes the time t_{th} it takes for the threshold potential to be reached and a spike to be produced, assuming that the neuron is in its reset state at $t = 0$, $v(0) = 0$. Writing down such an analytic expression is not possible for most – even slightly more complex – neuron models.

Computing the time-to-threshold t_{th} To compute t_{th}, we first need to solve the subthreshold differential equation under the assumption that J is constant and the initial condition is $v(0) = 0$. In general, this differential equation can be solved according to the "variation of constants" formula

$$v(t) = -\int_0^t \frac{1}{\tau_{\mathrm{RC}}}\big(v(t') - RJ\big)\,\mathrm{d}t' = RJ\left(1 - e^{-\frac{t}{\tau_{\mathrm{RC}}}}\right).$$

For the given assumptions, this equation can be used to compute the membrane potential at any time t. Of course, this does not take the super-threshold behaviour into account.

Second, we can compute the point in time t_{th} at which the membrane potential will reach v_{th}:

$$v_{\mathrm{th}} = RJ\left(1 - e^{-\frac{t_{\mathrm{th}}}{\tau_{\mathrm{RC}}}}\right),$$

$$\Leftrightarrow 1 - \frac{v_{\mathrm{th}}}{RJ} = e^{-\frac{t_{\mathrm{th}}}{\tau_{\mathrm{RC}}}}, \qquad\qquad \left| RJ \neq 0 \right.$$

$$\Leftrightarrow -\tau_{\mathrm{RC}} \log\left(1 - \frac{v_{\mathrm{th}}}{RJ}\right) = t_{\mathrm{th}}. \qquad\qquad \left| 1 - \frac{v_{\mathrm{th}}}{RJ} > 0, RJ \neq 0 \right.$$

Note: The terms right of the "|" specify the conditions for which equivalence holds. It is always useful to remember the conditions under which a derived equation remains valid.

Note that for $1 - \frac{v_{\mathrm{th}}}{RJ} < 0$ the input current is so small that the membrane potential will converge to an equilibrium below v_{th}, and hence $t_{\mathrm{th}} \to \infty$ – in other words, the neuron does not spike at all in this case.

Computing the firing rate   Note that the LIF neuron model assumes that neurons have no "memory". Whenever the neuron spikes, it is reset to it's initial state. This means that the LIF neuron response is perfectly periodic as long as the input remains constant.

> **Note:** The "no memory" assumption – and thus the perfect periodicity of the LIF neuron model – does not hold for other neuron models. However, for any neuron model, we could approximate $G[J]$ numerically by applying the current J, simulating the neuron over a time period T and then measuring the number of spikes n over T, i.e., $G[J] \approx n(J)/T$.

Given t_{th} we can compute the firing rate $G[J]$ of the neuron, i.e., the average number of output spikes during one second for a given input current J. The firing rate is the inverse of the inter-spike-interval (ISI), which is the sum of t_{th} (time-to-fire) and the refractory period τ_{ref}:

$$
G[J] = \begin{cases} \frac{1}{\tau_{\mathrm{ref}} + t_{\mathrm{th}}(J)} & \text{if } 1 - \frac{v_{\mathrm{th}}}{RJ} > 0, \\ 0 & \text{otherwise,} \end{cases}
$$

$$
= \begin{cases} \frac{1}{\tau_{\mathrm{ref}} - \tau_{\mathrm{RC}} \log\left(1 - \frac{v_{\mathrm{th}}}{RJ}\right)} & \text{if } 1 - \frac{v_{\mathrm{th}}}{RJ} > 0, \\ 0 & \text{otherwise.} \end{cases}
$$

Remember that we defined $v_{\mathrm{th}} = 1$. Furthermore, if we only want to characterize the LIF firing rate qualitatively, we can assume $R = 1$, since this resistance just rescales our input current (note that all variables are now unit-less). This gives us our final equation that fully summarizes the behaviour of the LIF neuron (fig. 4):

(LIF neuron rate approximation)

$$
G[J] = \begin{cases} \frac{1}{\tau_{\mathrm{ref}} - \tau_{\mathrm{RC}} \log\left(1 - \frac{1}{J}\right)} & \text{if } J > 1, \\ 0 & \text{otherwise.} \end{cases}
\tag{4}
$$

2.4   Limitations of the LIF neuron model

From looking at fig. 3, it may seem as if the LIF neuron model and the biophysically more plausible Hodgkin-Huxley type model exhibit the same qualitative behaviour. In general, this is not true – this observation only holds for certain sets of neuron parameters and inputs. While the LIF neuron model certainly is a good first-order approximation of neural behaviour, it does not account for some neurophysiological phenomena observed in biology (fig. 5).

For various reasons, we are not concerned with these limitations for now. First, as we will see when we talk about temporal representation, the methods we are using are not limited to LIF neurons – as long as we have a dynamical system description of the neuron we are using, we can compute the connection weights that implement a certain behaviour.

Second, before we add more complexity to our models – such as the ability to produce the complex neural responses from fig. 5 – we should ask ourselves first what the implications for producing high-level behaviour are. In our opinion, it only makes sense to add more detail if this somehow constrains the kind of behaviour our models can produce.

Lastly, many of these phenomena can also be produced by LIF neurons when considering network effects, and not just individual neurons.

> **Note:** As hinted at above, we will talk about more complex neuron models in the NEF at the end of the course, if time permits.

3   Artificial "Rate" Neurons

Using rate approximations instead of spiking dynamical systems is an idea that gave rise to the artificial neural networks used in machine learning – neurons are reduced to a function that maps an input (in biology: a current) onto a "firing rate". In machine learning, this function is called the "non-linearity". Being non-linear is an important aspect of computation in neural systems, since purely linear elements cannot perform any computation – apart from this, most artifical neuron response functions relatively little to do with actual neural response curves – more often than not, they are optimized for computationally cheap evaluation and differentiation instead of biological realism. We are mostly going over these nonlinearities as a point for comparison to the LIF rate approximation we just derived.

> **Note:** As a reminder, here is the definition of what it means for a function to be *linear*.
>
> A function $f : \mathbb{R}^m \longrightarrow \mathbb{R}^n$ is linear, *exactly if* the following equality holds
>
> $$f(a \cdot \mathbf{x} + b \cdot \mathbf{y}) = a \cdot f(\mathbf{x}) + b \cdot f(\mathbf{y}) \quad \text{for all } a, b \in \mathbb{R} \text{ and } \mathbf{x}, \mathbf{y} \in \mathbb{R}^m.$$
>
> Another definition that makes use of matrix multiplication is the following.
>
> A function $f : \mathbb{R}^m \longrightarrow \mathbb{R}^n$ is linear, *exactly if* there exists a matrix $\mathbf{F} \in \mathbb{R}^n \times \mathbb{R}^m$ such that
>
> $$f(\mathbf{x}) = \mathbf{F}\mathbf{x} \quad \text{for all } \mathbf{x} \in \mathbb{R}^m.$$
>
> This implies – reading the definition in the backwards direction – that every matrix $\mathbf{A}$ defines a corresponding linear function.

Next, we will take a look at some examples of neural non-linearities (or, in biological terms, "neural response functions").

The Rectified Linear Unit (ReLU)   This non-linearity is defined as $G[J] = \max\{0, J\}$ (fig. 6a). This is actually a reasonable substitute for the LIF rate approximation, especially in those cases where τ_{RC} is large (cf. fig. 4). Interestingly, this function has experienced a resurgence of interest with the recent developments in "deep learning".

The Softplus or Smooth ReLU function   This non-linearity is defined as $G[J] = \log(1 + \exp(J))$ (fig. 6b). The overall shape is the same as for the rectified linear unit, yet the sharp "kink" at $J = 0$ is smoothed out. In a neural setting, one finds such smoothing when superimposing noise onto the input current J. The Smooth ReLU function is hence vaguely comparable to rate approximations of the LIF neuron that take noise into account [**hunsberger2015spiking**].

The Logistic Activation function   This function is defined as $G[J] = \frac{1}{1+e^{-J}}$ (fig. 6c). It is a so called *sigmoid* function (because of the "S" shape). If we wanted to interpret this activation function from a biological standpoint, we could say that this function captures the smooth onset observed in a noisy system, as well as the saturation effect we see in the LIF rate approximation. The Logistic Activation Function and the Hyperbolic Tangent function are among the most widely used non-linearities in machine learning (at least before the ReLU took over).

The Hyperbolic Tangent function   This function is defined as $G[J] = \tanh(J) = \frac{e^{J}-e^{-J}}{e^{J}+e^{-J}}$ (fig. 6d) and is similar in shape to the logistic activation function – one difference to all the other functions we looked at so far is that we have negative rates – this is something that matters very little in machine learning, but would definitively not be a valid response function for our purposes. This emphasizes again that artifical neurons are not at all designed with biological plausibility in mind.
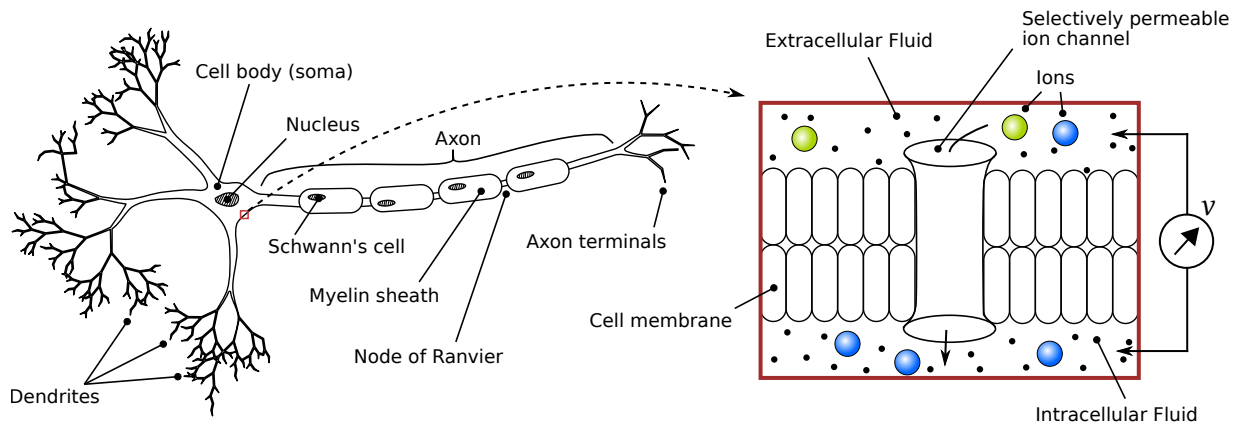
Figure 1: Illustration showing a text-book neuron, as well as a schematic cross-section through the cell membrane. Left part of the illustration from [**stoeckel2015design**], adapted from [**kandel2012principles**].


(a) Sub-threshold neuron

(b) Super-threshold neuron
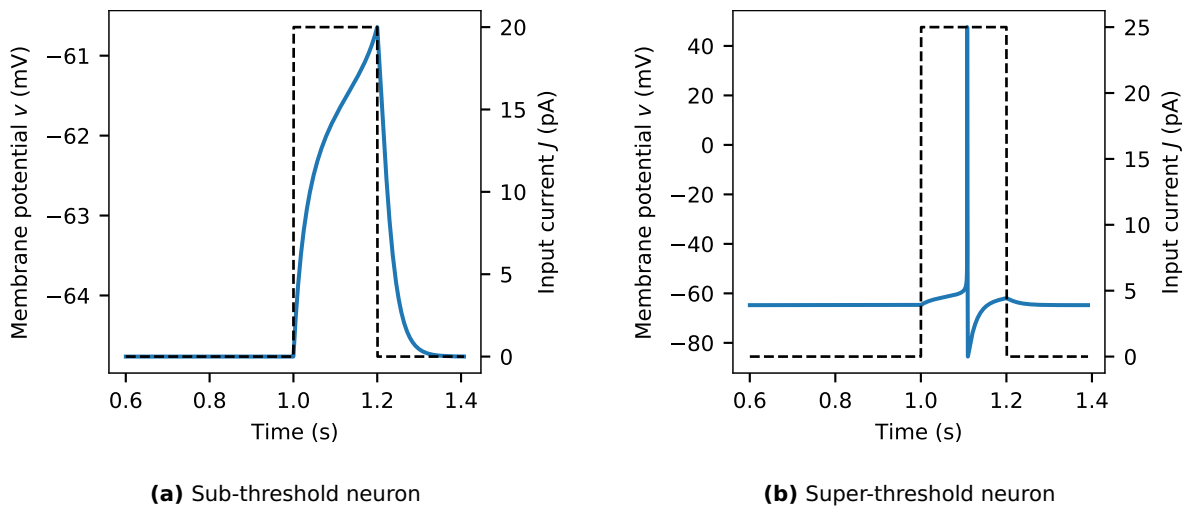
Figure 2: Computer simulation of a Hodgkin Huxley type model neuron [**hodgkin1952quantitative**, **traub1991neuronal**]. The blue line corresponds to the membrane potential v, the dashed line to the current J that is being injected into the neuron. The time axis starts at 0.6 s to make sure that the neuron has settled into its resting state. ▦ *Code (simulation)*, ▦ *Code (plotting)*
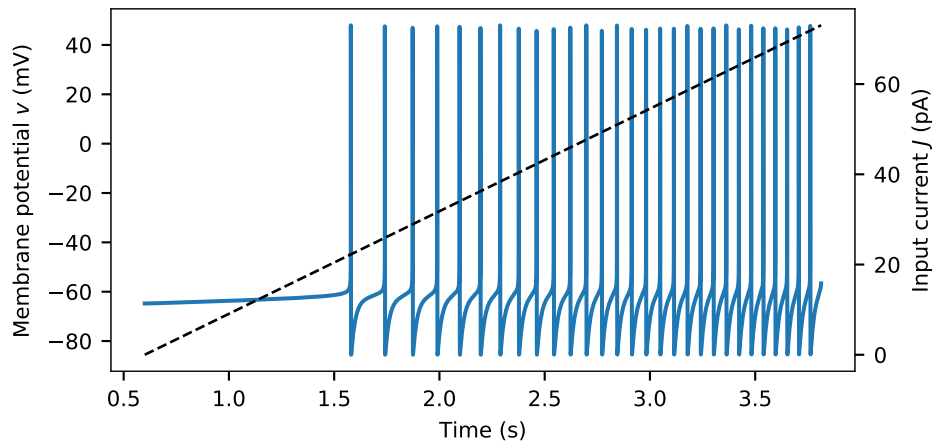
(a) Hodgkin-Huxley model neuron
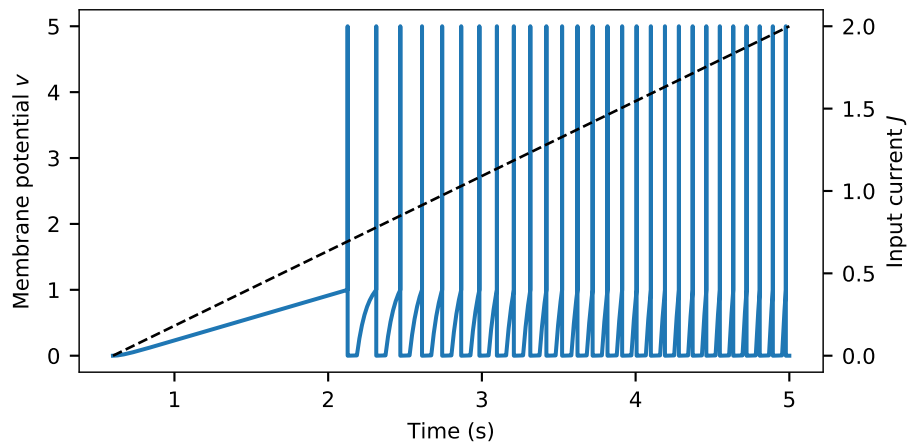
(b) Normalized Leaky Integrate-and-Fire neuron

Figure 3: Effects of a current ramp on an Hodgkin-Huxley type model neuron and a (normalized) LIF neuron. As above, the blue line is the membrane potential, the dashed line is the input current. ▨ *Code (simulation)*, ▨ *Code (plotting)*
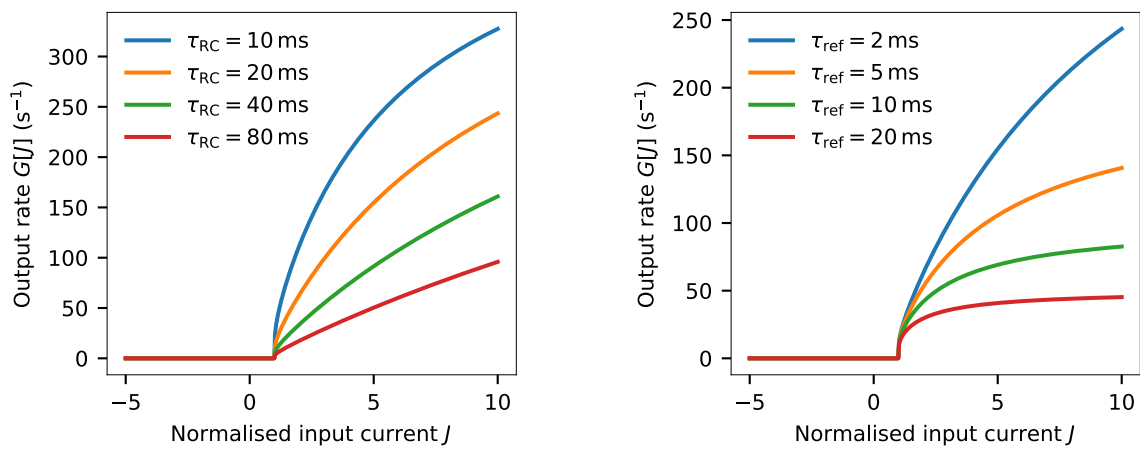
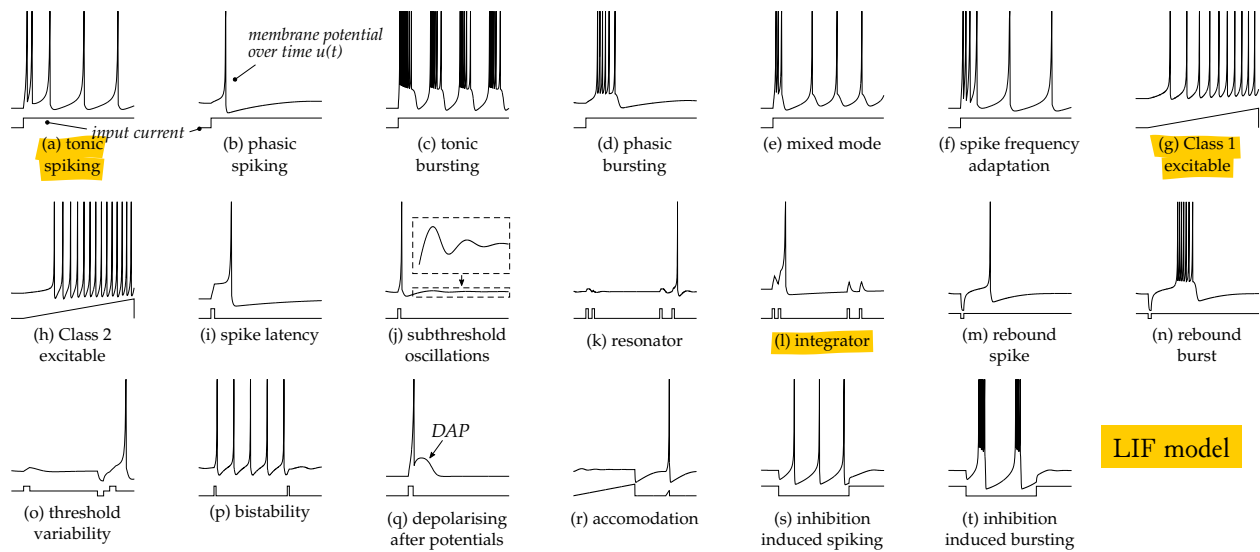**Figure 4:** LIF rate approximation response curve $G[J]$ for different τ_{RC} and τ_{ref}. ▦ *Code*


Figure 5: Single-neuron behaviours observed in nature. The highlighted behaviours correspond to those that are explained by the LIF neuron model. Upper plot of each subfigure corresponds to the membrane potential, lower plot to the input current. Figure from [**stoeckel2015design**], adapted from [**izhikevich2004model**]. All these behaviours can be explained with the slightly more complex (two instead of one state variables) Izhikevich neuron model [**izhikevich2003simple**].
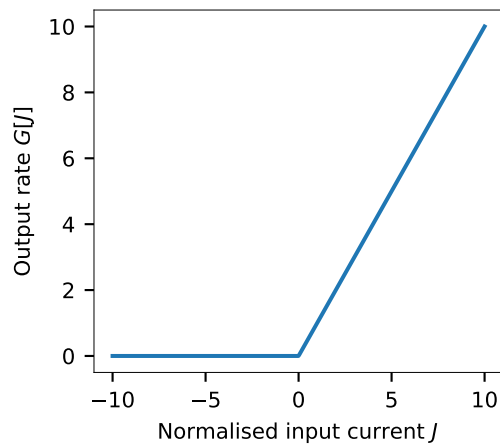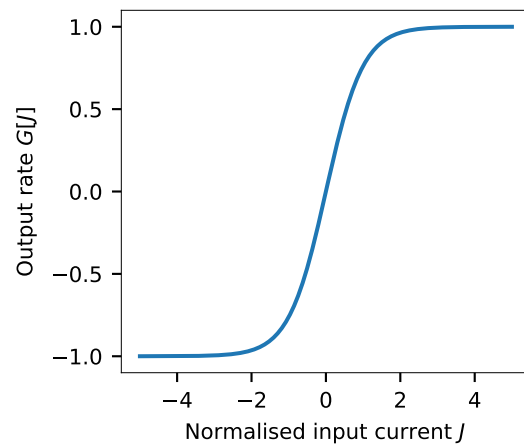
(a) Rectified Liner (ReLU)
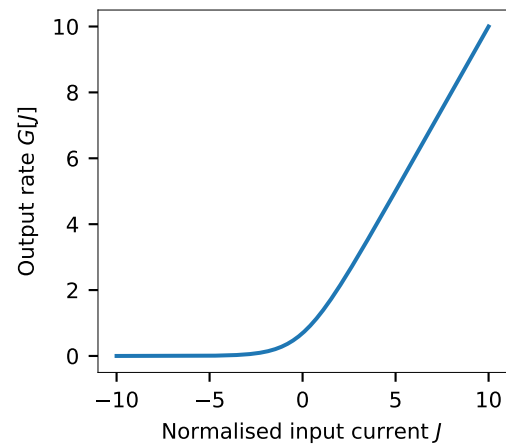
(b) Soft Rectified linear
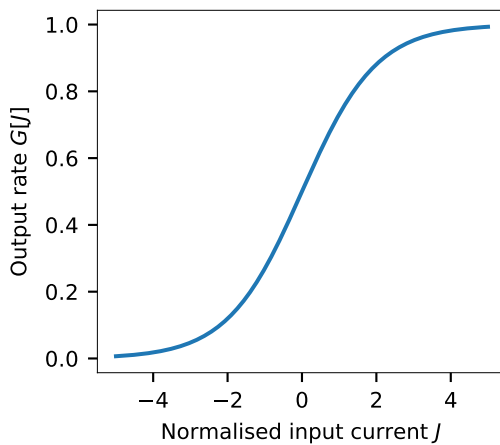
(c) Logistic function

(d) Hyperbolic Tangent

Figure 6: Overview of some neural response functions used in artifical neural networks. ▨ *Code*