Neuroprothetik – Exercise 3: Mathematical Basics II

1 Implementations in Matlab or Python

1.1 Forward (Explicit) Euler

The forward Euler has been implemented as a function in Matlab (`function ... [V, t] = forwardEuler(h, N, DGL, V_0, t_0)`). The input variables are the step length `h`, the maximum time length `N`, a function handle to a differential equation `DGL`, starting value for the voltage `V_0` and the starting point for the time vector `t_0`. *See code for more detailed information.*

1.2 Heun Method

The Heun method has been implemented as a function in Matlab (`function ... [V, t] = heunMethod(h, N, DGL, V_0, t_0)`). The input variables have been described previously. *See code for more detailed information.*

1.3 Exponential Euler

The exponential Euler has been implemented as a function in Matlab (`function ... [V, t] = exponentialEuler(h, N, A, B, V_0, t_0)`). The input variables are basically the same as stated before, except for the function handle `DGL`, which is now divided into to function handles `A` and `B`, as presented in the lecture. *See code for more detailed information.*

2 Solve Functions

In this section the differential equation from the previous problem sheet will be solved:

$$\frac{dV}{dt} = 1 - V - t. \tag{1}$$

The starting point is $V(t_0) = V(-4.5) = V_0 = -4$. The step size will be $h \in \{1s, 0.5s, 0.1s, 0.012s\}$.

2.1 Forward (Explicit) Euler

The results for the *forward Euler* are shown in fig. 1. The larger the step size, the bigger is the overshoot at the local maximum for that function. Smaller step sizes yield better results, i.e. approximate the curve much better. For $h = 1s$ there is an undershoot, but still the curve looks better

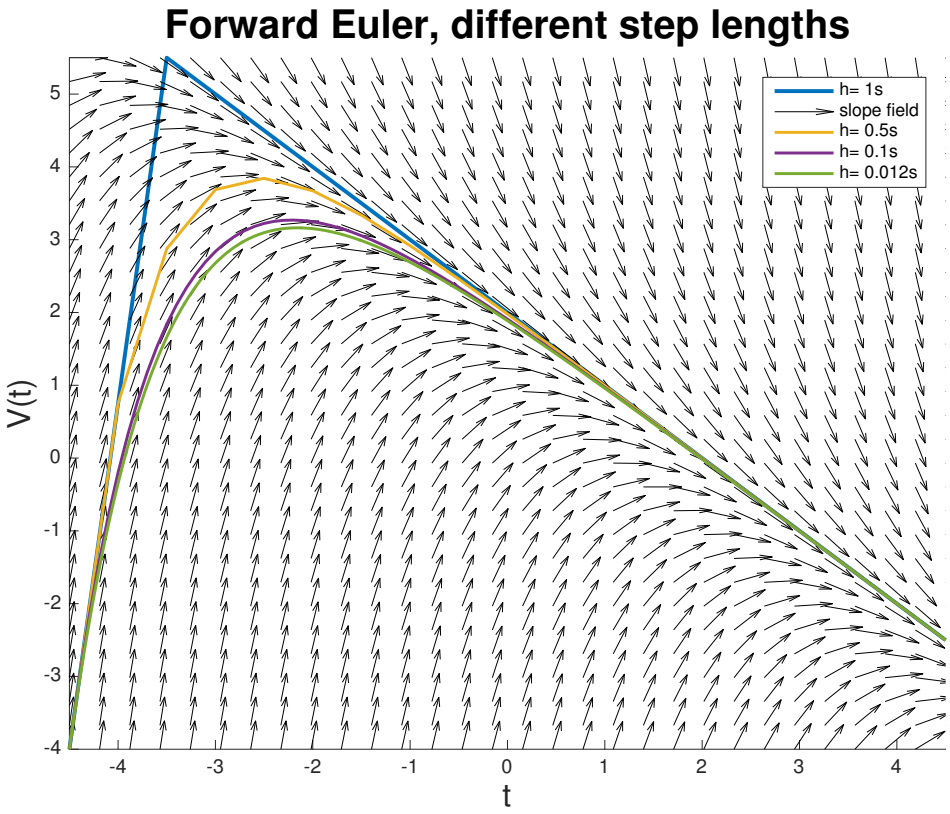

Figure 1: Forward Euler with different step sizes.

2.2 Heun Method

The results for the *Heun method* are shown in fig. 2. When compared to the forward Euler, the Heun method converges faster. The curves for $h = 0.012s$ and $h = 0.1s$ even overlap completely in the figure. Here, the low step size $h = 1s$ yields an undershoot, but looks smoother overall.

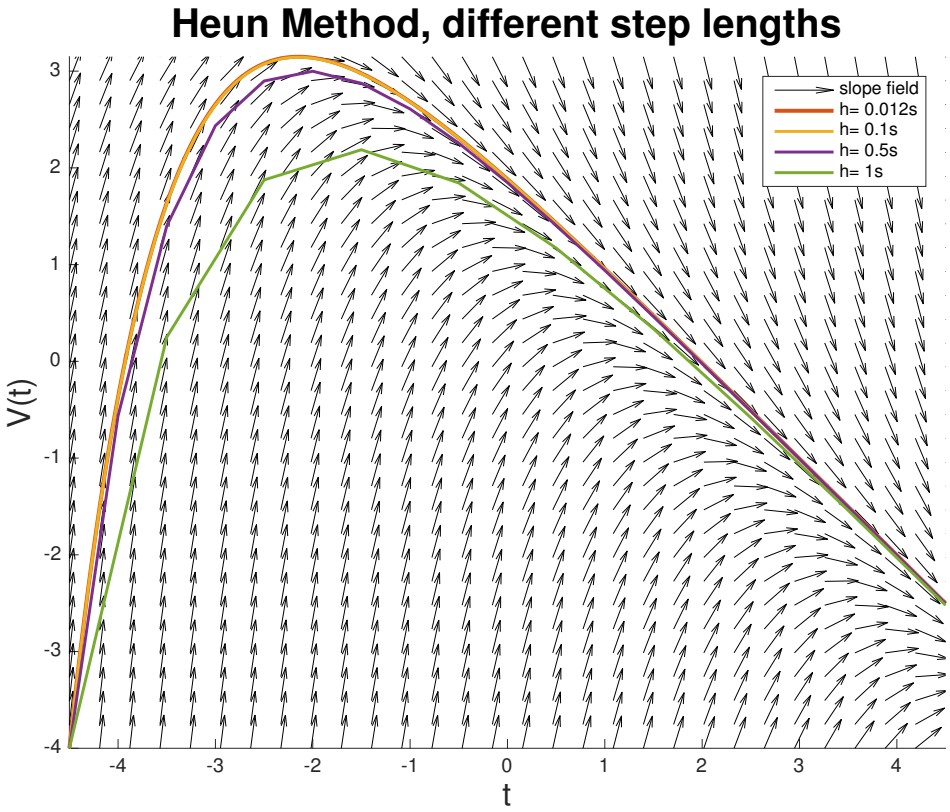

Figure 2: Heun method with different step sizes. The plots for $h = 0.012s$ and $h = 0.1s$ are overlapping and thus not well distinguishable.

2.3 Exponential Euler

The results for the *Exponential Euler* are shown in fig. 3. The exponential Euler exhibits a constant offset with large step sizes (e.g. `h = 1s`). This offset gets smaller with increasingly small step sizes. However, the approximation looks the best overall when neglecting the offset for the bigger step sizes.

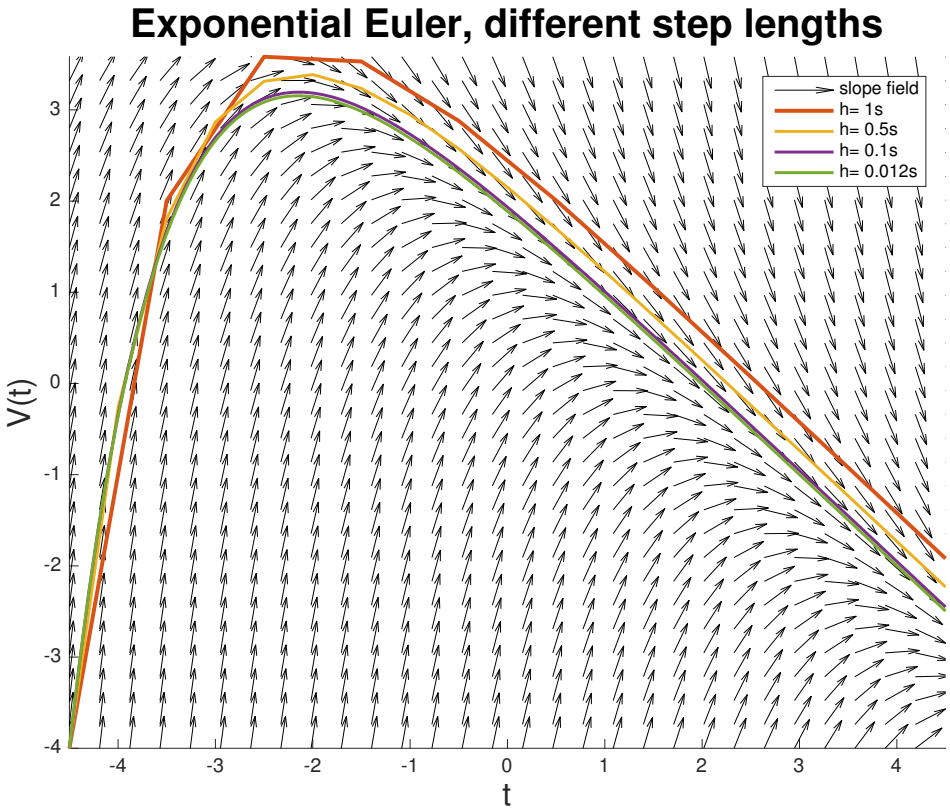

Figure 3: Exponential Euler with different step sizes. The plots for $h = 0.012s$ and $h = 0.1s$ are nearly overlapping and hardly distinguishable.

Why not just use a very small step size?

The explicit Euler method is a first-order method, which means that the local error (error per step) is proportional to the square of the step size, and the global error is proportional to the step size. Using a small step size thus results in a better approximation and a smaller error. However, a small step size also means a large number of steps in total, which entails a *high computational cost*. Furthermore, for rapidly changing functions, a small step size could even increase the error. Therefore, other approximations, such as the Runge-Kutta methods are needed.

3 The Leaky Integrate and Fire Neuron

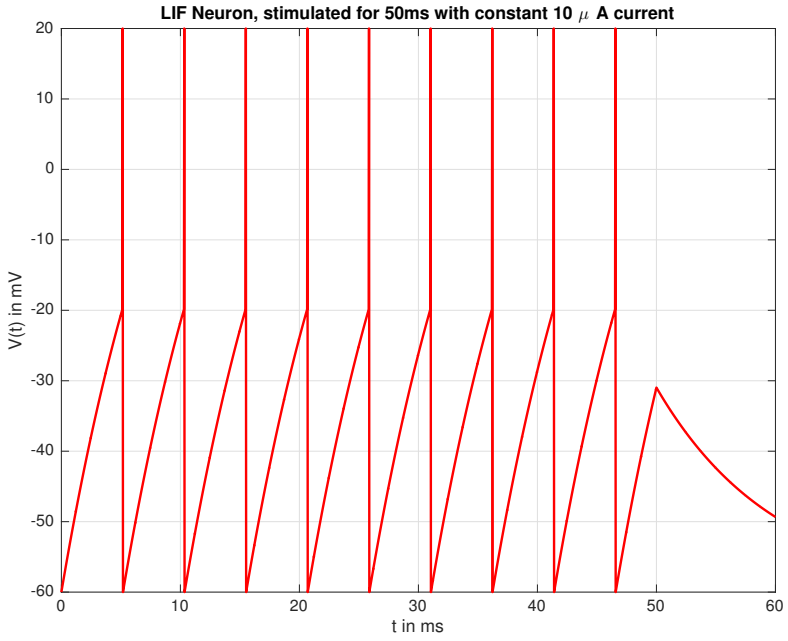

Figure 4: LIF neuron stimulated for 50 ms with constant current $I = 10\mu$A

Interpretation of the results

The *leaky integrate and fire* (LIF) neuron is mere passive description of the membrane. The current that is applied to the neurons may be thought of as a patch electrode inserted into the neuron.
The neuron model integrates over the input current by charging the membrane capacitance. When reaching a threshold voltage of $V_m = -20$ mV, the neuron spikes to $V_{spike} = 20$ mV (e.g. see figure 4, where this is nicely visible). After having spiked, the neuron's membrane potential is reset to $V_m = V_{rest} = -60$ mV, and the integration of the persistent input current reoccurs, resulting in a recurrent spiking in fixed intervals (see figure 4 for example). When comparing figures 4 and 5, we notice that the neuron spikes more often when stimulated with a higher current (10μA vs. 20μA). This model sets no limits in spiking frequency, which is unrealistic. A real neuron has a refractory time of at least 1 to 2 ms, so the maximum spiking frequency response would be between 500 and 1000 Hz.
The same holds for figures 6 and 7, where a rectified sine was applied. However, here we see *bumps* in the integration phase. This is due to the input current going very low periodically due to its sine nature, and the *leaky* part (discharging of the membrane capacitance) of the model is stronger than the charging of the membrane capacitance for a very short time. When the sine rises again up to its amplitude value, the membrane capacitance potential reaches the threshold value

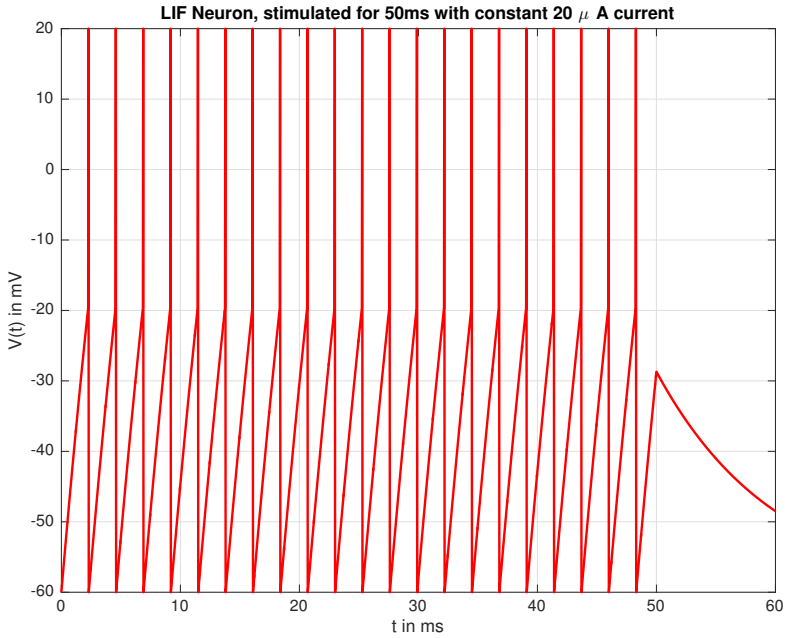

Figure 5: LIF neuron stimulated for 50 ms with constant current $I = 20\mu$A

and the neuron spikes a several times, while the integrated current is still high enough.
Edit: The time window of the plots is between 0 and 60 ms, whereas the neuron is only stimulated for 50 ms. That way, one can better see the discharging of the membrane capacitance through the leak conductance.

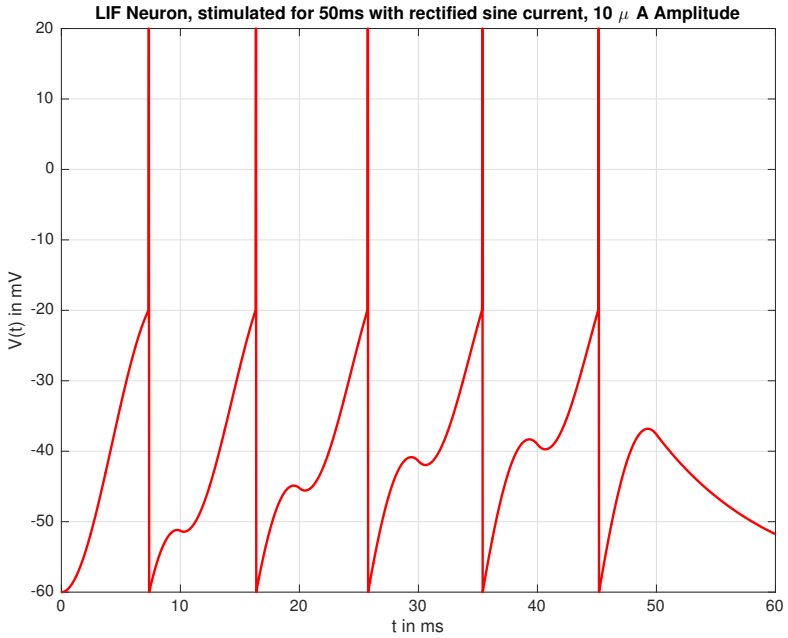

Figure 6: LIF neuron stimulated for 50 ms with rectified sine current, $A = 10\mu$A Amplitude.

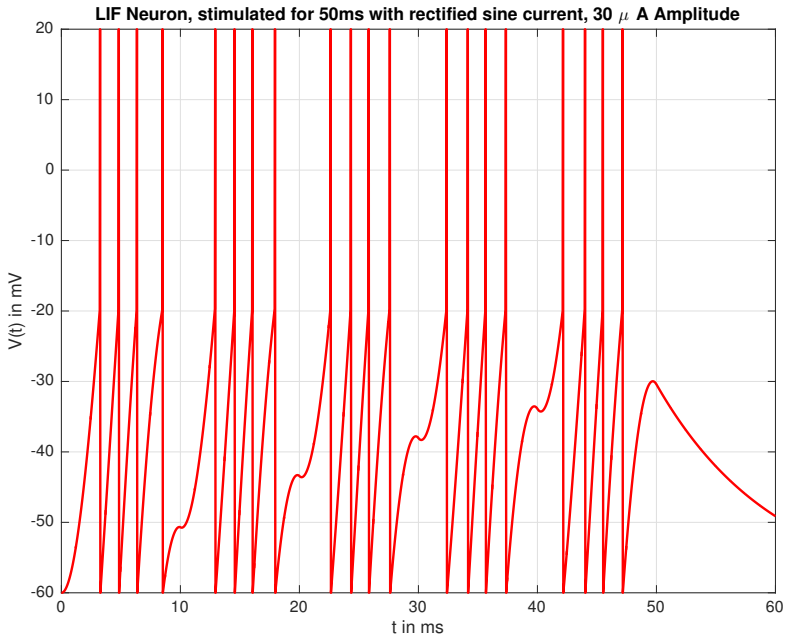

Figure 7: LIF neuron stimulated for 50 ms with rectified sine current, $A = 30\mu$A Amplitude.