The Leaky Integrate-and-Fire Neuron Model

Emin Orhan
eorhan@bcs.rochester.edu

November 20, 2012

In this note, I review the behavior of a leaky integrate-and-fire (LIF) neuron under different stimulation conditions. I closely follow chapter 4.1 of Gerstner and Kistler (2002). I consider three different stimulation conditions below and show how each condition can be implemented in the Brian spiking neural network simulator for Python (Goodman & Brette, 2008).

Introduction: The leaky integrate-and-fire (LIF) neuron is probably one of the simplest spiking neuron models, but it is still very popular due to the ease with which it can be analyzed and simulated. In its simplest form, a neuron is modeled as a "leaky integrator" of its input $I(t)$:

$$\tau_m \frac{dv}{dt} = -v(t) + RI(t) \tag{1}$$

where $v(t)$ represents the membrane potential at time t, τ_m is the membrane time constant and R is the membrane resistance. This equation describes a simple resistor-capacitor (RC) circuit where the leakage term is due to the resistor and the integration of $I(t)$ is due to the capacitor that is in parallel to the resistor. The spiking events are not explicitly modeled in the LIF model. Instead, when the membrane potential $v(t)$ reaches a certain threshold v_{th} (spiking threshold), it is instantaneously reset to a lower value v_r (reset potential) and the leaky integration process described by Equation 1 starts anew with the initial value v_r. To add just a little bit of realism to the dynamics of the LIF model, it is possible to add an absolute refractory period Δ_{abs} immediately after $v(t)$ hits v_{th}. During the absolute refractory period, $v(t)$ might be clamped to v_r and the leaky integration process is re-initiated following a delay of Δ_{abs} after the spike.

1. Stimulation by a constant input current: Consider the case of constant input: $I(t) = I$. We assume $v_r = 0$. The solution of Equation 1 is then given by:

$$v(t) = RI[1 - \exp(-\frac{t}{\tau_m})] \tag{2}$$

It is easy to understand the behavior of this solution. The asymptotic value of the membrane potential is RI. If this value is less than the spiking threshold, v_{th}, no spike can be generated. If, however, $RI > v_{th}$, then the neuron generates spikes ("fires") periodically. Assuming $v(0) = v_r = 0$, the time of the first spike, $t^{(1)}$, can be found by solving:

$$v_{th} = RI[1 - \exp(-\frac{t^{(1)}}{\tau_m})] \tag{3}$$

This yields:

$$t^{(1)} = \tau_m \ln \frac{RI}{RI - v_{th}} \tag{4}$$

Equation 4 also gives the time between each successive spike the neuron fires (or its period). This is because we assumed $v(0) = v_r$ in deriving Equation 4. If we also add an absolute refractory period, Δ_{abs}, as discussed above, the period becomes:

$$T = \Delta_{abs} + \tau_m \ln \frac{RI}{RI - v_{th}} \tag{5}$$

The mean firing rate of the neuron, f, is then given by $1/T$:

$$f = [\Delta_{abs} + \tau_m \ln \frac{RI}{RI - v_{th}}]^{-1} \tag{6}$$

Figure 1 (left) shows the simulation of a LIF neuron with a constant input current. In this simulation, we take $v_r = 0$ mV, $v_{th} = 15$ mV and stimulate the neuron with a suprathreshold constant current of $I = 20$ mV (the current is given in units of mV, because we assume $R = 1$ mΩ; thus, a more accurate way of expressing this would be $RI = 20$ mV, but for the sake of brevity we will simply omit R in what follows). As predicted, the neuron fires regularly. Figure 1 (right) shows the $f - I$ (mean firing rate-input current) curves for a LIF neuron with (dashed line) and without (solid line) an absolute refractory period. These curves are calculated analytically using Equation 6. The following Brian code shows how to simulate a LIF neuron with a constant input current:

```
from brian import *
from numpy import *

if __name__ == '__main__':

    N = 1  # number of neurons

    tau_m = 10 * ms # membrane time constant
    v_r   = 0 * mV  # reset potential
    v_th  = 15 * mV # threshold potential
    I_c   = 20 * mV # constant input current

    eqs = '''
    dv/dt = -(v-I)/tau_m : volt
    I : volt
    '''

    lif   = NeuronGroup(N, model=eqs, threshold=v_th, reset=v_r)
    # You can add randomness in initial membrane potential by changing the following line
    lif.v = v_r * mV + 0 * mV * rand(len(lif))
    lif.I = I_c

    spikes  = SpikeMonitor(lif)
    v_trace = StateMonitor(lif, 'v', record=True)

    run(0.1*second)

    figure(1)
    plot(v_trace.times/ms,v_trace[0]/mV)
    xlabel('Time (ms)', fontsize=24)
```

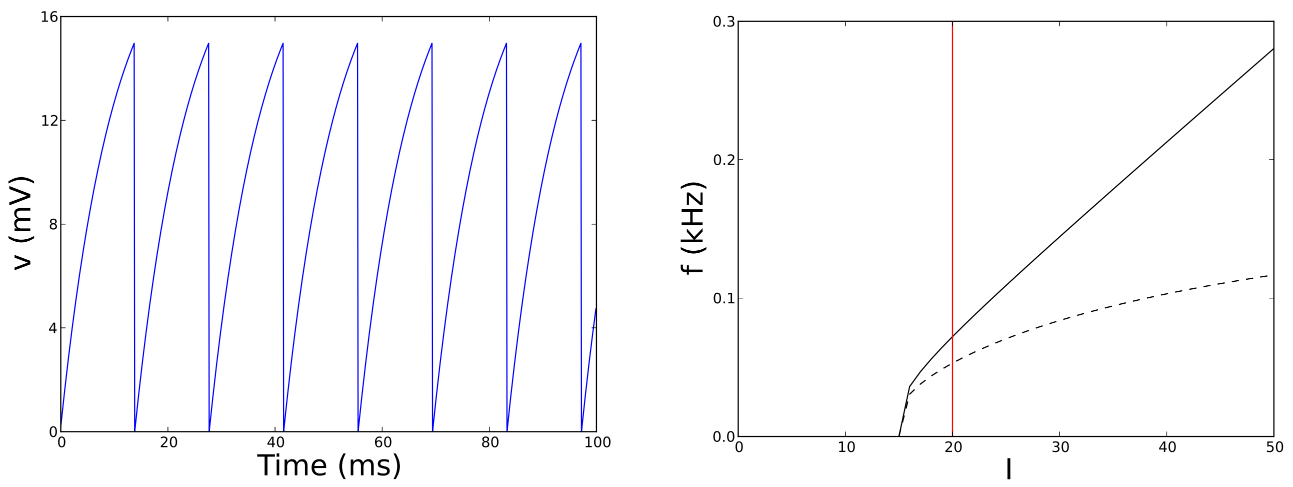

Figure 1: Stimulation of a LIF neuron by a constant input current: the time-course of the membrane potential, $v(t)$ (left); f-I curve for a LIF neuron with (dashed line: $\Delta_{abs} = 5$ ms) and without (solid line: $\Delta_{abs} = 0$ ms) an absolute refractory period (right). Red line indicates the particular I value used in the simulation on the left.

```
    ylabel('v (mV)',fontsize=24)
    yticks([0,4,8,12,16])

    show()

    # plot the f-I curve for a LIF neuron with constant input
    delta_abs = 5
    tau_m = 10
    v_th = 15
    Is = linspace(0,50,51)

    f1 = 1.0 / (0 + tau_m * log(Is/(Is - v_th)))
    f2 = 1.0 / (delta_abs + tau_m * log(Is/(Is - v_th)))

    figure(2)
    plot(Is,f1,'k-')
    plot(Is,f2,'k--')
    plot([20, 20],[0, 0.3],'r-')
    xlabel('I',fontsize=24)
    ylabel('f (kHz)',fontsize=24)
    yticks([0,.1,.2,.3])

    show()
```

2. Stimulation by a time-varying input current: For a general time-varying input current $I(t)$, the solution of Equation 1, with the initial condition $v(t_0) = v_r$, is given by:

$$v(t) = v_r \exp(-\frac{t - t_0}{\tau_m}) + \frac{R}{\tau_m} \int_0^{t-t_0} \exp(-\frac{s}{\tau_m}) I(t - s) ds \tag{7}$$

As in the previous case, this equation describes the dynamics of the membrane potential between successive

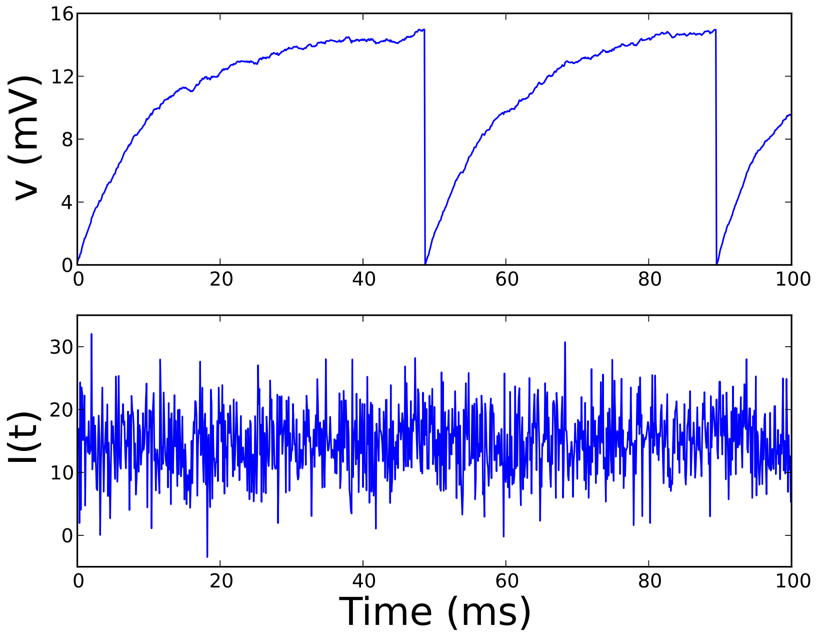

Figure 2: Stimulation of a LIF neuron by a time-varying input current. The lower panel shows the time-varying input current, $I(t)$. In this example, at each time step t, $I(t)$ is sampled from a Gaussian with mean 15 and standard deviation 5. Note that we take $R = 1$ mΩ, so the input current I and the membrane potential v have the same scale and the same units. Other parameters are the same as in the previous simulation: in particular $v_r = 0$, $v_{th} = 15$ mV. The upper panel shows the time-course of the membrane potential, $v(t)$. Note that spiking is no longer regular.

spiking events. When the membrane potential reaches the threshold, it is instantaneously reset to v_r and starts to evolve according to Equation 7 again until the next spiking event. It is a simple but very instructive exercise to show that Equation 2 emerges as a special case of Equation 7 when the input current is constant: i.e. $I(t) = I$ (note that we assumed $v_r = 0$ and $t_0 = 0$ in deriving Equation 2).

Figure 2 shows the simulation of a LIF neuron with a time-varying input current, $I(t)$. In this example, at each time step t, $I(t)$ is sampled from a Gaussian distribution with mean 15 mV and standard deviation 5 mV (again we omit R here, so the input current $I(t)$ is specified in units of mV). This condition can be simulated using the same Brian code given above. The only thing that needs to be changed is line 21: here, we should substitute a time-varying current for the constant current used in the previous simulation. One way to do this in Brian is to use a `TimedArray`:

```
lif.I = TimedArray((15 + 5*randn(1000))*mV)
```

This specifies the input current at each time-step of the simulation. We need to specify 1000 values in total, because our total simulation time is 100 ms and each time-step in the simulation is 0.1 ms, thus the total number of time-steps in the simulation is 1000. The resulting time-varying input current is shown in the lower panel of Figure 2 and the time-course of the membrane potential of the LIF neuron is shown in the upper panel of Figure 2.

3. Stimulation by synaptic currents: Previous sections considered the stimulation of the LIF neuron by the direct injection of constant or time-varying currents. Now, consider a more realistic situation where the neuron is stimulated by pre-synaptic spikes arriving at its synapses. Each pre-synaptic spike makes a stereotyped contribution, described by a function $\alpha(t)$, to the post-synaptic current and contributions of different pre-synaptic spikes are linearly summed to obtain the total post-synaptic current. Thus, the

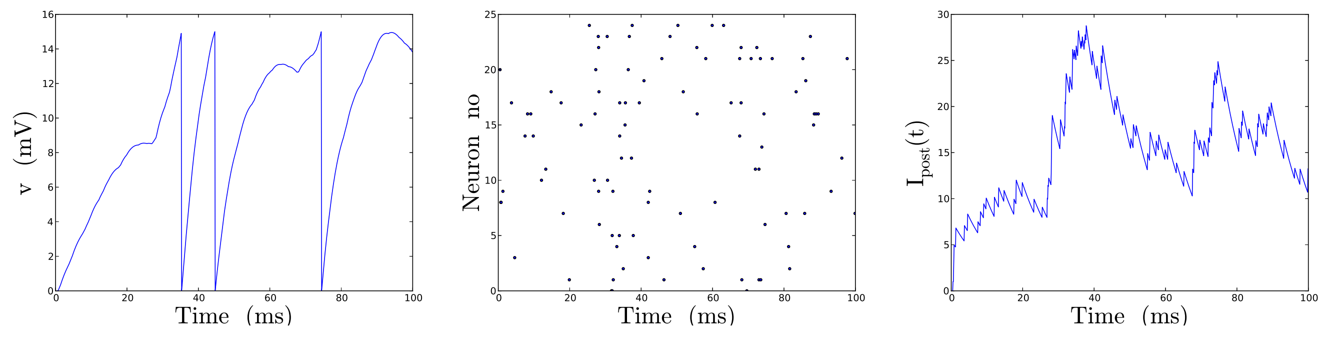

Figure 3: Stimulation of a LIF neuron by synaptic currents: the time-course of the post-synaptic membrane potential, $v(t)$ (left); raster plot of pre-synaptic spikes (middle); the total post-synaptic current, $I_{post}(t)$, due to pre-synaptic spikes (right).

total post-synaptic current to the i-th neuron can be expressed as follows:

$$I_i(t) = \sum_j w_{ij} \sum_f \alpha(t - t_j^{(f)}) \tag{8}$$

where $t_j^{(f)}$ represents the time of the f-th spike of the j-th pre-synaptic neuron; w_{ij} is the strength of synaptic efficacy between neuron i and neuron j. Common choices for α include (1) the instantaneous Dirac δ-pulse:

$$\alpha(t) = q\delta(t) \tag{9}$$

where q is the total charge injected into the synapse; (2) the alpha synapse (I use the expressions in Brian's documentation below, Gerstner & Kistler (2002) use slightly different expressions for these synapses):

$$\alpha(t) = \alpha \frac{t}{\tau} \exp(1 - \frac{t}{\tau}) \tag{10}$$

or (3) the bi-exponential synapse:

$$\alpha(t) = \beta \frac{\tau_2}{\tau_2 - \tau_1} [\exp(-\frac{t}{\tau_1}) - \exp(-\frac{t}{\tau_2})] \tag{11}$$

where α and β are normalizing constants and τ, τ_1 and τ_2 are the time constants of the synapses.

Figure 3 shows the simulation a LIF neuron with 25 pre-synaptic inputs arriving at an alpha synapse (Equation 10). In this example, we modeled the pre-synaptic inputs as independent Poisson processes with a common rate of 40 Hz. The synaptic efficacies w_{ij} (see Equation 8) were randomly drawn from a Gaussian distribution with mean 1.62 mV and standard deviation 0.5 mV. Other parameters of the LIF neuron were the same as in previous simulations (in particular, $v_r = 0$, $v_{th} = 15$ mV).

The following Brian code shows how to perform the simulation demonstrated in Figure 3. Here, we used the `alpha_synapse` function (line 18 below) from the `brian.library.synapses` module which contains a few useful built-in synapses such as the alpha synapse and the bi-exponential synapse. For a bi-exponential synapse, we would use the `biexp_synapse` function from the same module: i.e. something like `biexp_synapse(input='I_in',tau1=10*ms,tau2=5*ms,unit=mV,output='ge')` on line 18 below. For a Dirac δ-pulse synapse, simply removing the line 18 entirely would be enough (why?).

```
1  from brian import *
2  from brian.library.synapses import alpha_synapse
3  from brian.library.IF import leaky_IF
```

```
from numpy import *

if __name__ == '__main__':

    N_pre  = 25    # number of presynaptic neurons
    N_post = 1     # number of postsynaptic neurons

    tau_m = 10 * ms # membrane time constant
    v_r   = 0 * mV  # reset potential
    v_th  = 15 * mV # threshold potential
    W     = 1.62 * mV + 0.5*randn(N_pre,N_post) * mV # synaptic efficacies

    # Leaky IF neuron with alpha synapses
    eqs = leaky_IF(tau=tau_m,El=v_r)+Current('I=ge:mV')+\
          alpha_synapse(input='I_in',tau=10*ms,unit=mV,output='ge')

    lif_pre    = PoissonGroup(N_pre, rates=40*Hz)
    lif_post   = NeuronGroup(N_post, model=eqs, threshold=v_th, reset=v_r)
    C          = Connection(lif_pre, lif_post, 'ge', weight=W)

    spikes_pre   = SpikeMonitor(lif_pre)
    spikes_post  = SpikeMonitor(lif_post)
    v_trace      = StateMonitor(lif_post, 'vm', record=True)
    I_trace      = StateMonitor(lif_post, 'ge', record=True)

    run(0.1*second)

    figure(1)
    plot(v_trace.times/ms,v_trace[0]/mV)
    xlabel('$\mathrm{Time \; (ms)}$',fontsize=30)
    ylabel('$\mathrm{v \; (mV)}$',fontsize=30)

    figure(2)
    plot(I_trace.times/ms,I_trace[0]/mV)
    xlabel('$\mathrm{Time \; (ms)}$',fontsize=30)
    ylabel('$\mathrm{I_{post}(t)}$',fontsize=30)

    figure(3)
    raster_plot(spikes_pre)
    xlabel('$\mathrm{Time \; (ms)}$',fontsize=30)
    ylabel('$\mathrm{Neuron \; no}$',fontsize=30)
    show()
```

References

[1] Gerstner, W. & Kistler, W. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge University Press.

[2] Goodman, D. & Brette, R. (2008). Brian: a simulator for spiking neural networks in Python. *Frontiers in Neuroinformatics*, **2**:5. doi: 10.3389/neuro.11.005.2008.