Problem Set: The Integrate-and-Fire Model

Matteo Vissani
PhD Course: Information Theory and Neural Modelling
Scuola Superiore Sant'Anna

May 7, 2020

In this exercise we want to simulate a simple model of a neuron called Integrate-and-Fire model. The neuron accumulates the total incoming current in the soma until a voltage threshold is reached. This will trigger an action potential thereby resetting the membrane potential back to its resting potential. Because the cell membrane of the neuron is leaky, the neuron will lose its accumulated charge over time. Therefore the neuron will not fire if the accumulation of charge is too slow. If we neglect for a moment that the neuron will fire an action potential after reaching a voltage threshold, the change in membrane voltage $V(t)$ can be described by the following differential equations [1, 2, 3]:

$$\underbrace{C\frac{\mathrm{d}V(t)}{\mathrm{d}t}}_{\text{Membrane}} + \underbrace{g_L(V(t)-E_L)}_{\text{Leakage}} - \underbrace{g_L D_T e^{\frac{V(t)-V_T}{D_T}}}_{\text{Exponential}} + \underbrace{w(t)}_{\text{Adaptation}} + \underbrace{g_{Ext}s(t)(V(t)-E_E)}_{\text{Input}} = 0 \tag{1}$$

$$\underbrace{\tau_w\frac{\mathrm{d}w(t)}{\mathrm{d}t} + w(t) = a(V(t)-E_L) + \overbrace{\sum_{i=0}^{n} b\delta_(t-t_i)}^{\text{Neural activity}}}_{\text{Adaptation dynamics}} \tag{2}$$

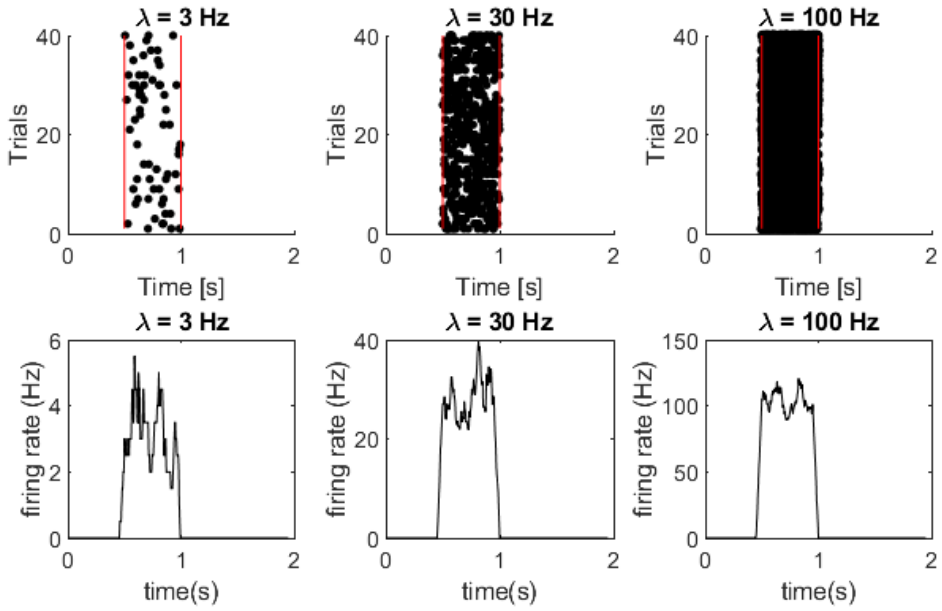
$$\underbrace{\tau_s\frac{\mathrm{d}s(t)}{\mathrm{d}t} + s(t) = \overbrace{\sum_{j=0}^{m}\delta_{Ext}(t-t_j)}^{\text{External activity}}}_{\text{Synapses Dynamics}} \tag{3}$$

Once the neuron reaches the threshold V_p (i.e. $V(t) \geq V_p$), the membrane voltage is kept at a resting value V_r for the duration of its absolute refractory time T_{abs}

$$V(t) \longleftarrow V_r; \quad t_j \leq t \leq t_j + T_{abs} \tag{4}$$

Numerical Implementation

The code is composed by one main script and three auxiliary functions:
- *Main IaF Neuron.mat* is the main script that handles the whole simulation. Before calling *aEIF.mat*, the script builds the external stimulus delivered by a single excitatory synapse for a certain time period.




```
% Stimulus Parameter
% Time Parameter
I_start = 0.5; % stimulus onset in s
I_start_step = floor(I_start/h);
I_stop = 1; % stimulus offset in s
I_stop_step = floor(I_stop/h);
I_length = I_stop - I_start; % stimulus length
I_length_step = I_stop_step - I_start_step + 1;
% Frequency Parameter
v_ext = 300*ones(1,N)'; % stimulus frequency
```

- *spike2rate.mat* is a function that converts the binary spike train in a continuous firing rate FR(t) through a convolution with a rectangular kernel K(t) characterized by a window W_k (see [4] for a review of methods):

$$FR(t) = \int_0^t \sum_{j=0}^{m} \delta_V(\tau - t_j) K(t-\tau) d\tau \tag{5}$$

$$K(t-\tau) = \frac{1}{W_k}; \quad |t-\tau| \leq \frac{W_k}{2} \tag{6}$$

```
function rate=spike2rate(spike_count,dt,time_window,time_slide)
% ####################################
% Author:  M. Vissani, 2018
% ####################################
    slide_steps = floor(time_slide/dt);
    length_steps = floor(length(spike_count)/slide_steps);
    windows_steps= floor(floor(time_window/dt)/slide_steps);
    spike_hist_tmp = spike_count(1:(length_steps*slide_steps));
    spike_hist_tmp = reshape(spike_hist_tmp,slide_steps,length_steps);
    spike_hist = sum(spike_hist_tmp);
    kernel = zeros(1,length_steps);
    kernel(1:windows_steps) = 1;
    rate_tmp = conv(spike_hist,kernel);
    rate = rate_tmp(windows_steps:length_steps)/time_window;
end
```

- *generate-ext-poisson.mat* is a function that generates a Poissonian spike train with an expected value λ (see [5] for more statistical details).

$$P_\lambda(n) = \frac{\lambda^n}{n!} e^{(-\lambda)} \tag{7}$$

```
function ext_input=generate_ext_poisson(v_ext_eff,N,dt, steps,I_start_step, I_stop_step, I_length_step)
% ###################################
% Author:  M. Vissani, 2018
% ###################################
    ext_input = zeros(N,steps);
    ext_input(1:N,I_start_step : I_stop_step)=(rand(N,I_length_step) < ones(N,I_length_step).*(v_ext_eff*dt));
end
```

- *aeIF.mat* is the computational core that extracts the dynamics of the neuron. Differential equations are numerically solved using a Forward Euler Scheme with a step of integration h. Ensure that the condition of numerical stability $\left|1 + \frac{hg_{Ext}}{C}\right| < 1$ is satisfied, otherwise you will get funny results (see [6] for more details).

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = f(t, y(t), x(t)) \Rightarrow y_{n+1} = y_n + hf(t_n, y_n, x_n) \tag{8}$$

```
function [V, w ,St,Sb, I_ext] = aEIF(t,I,par)

% Author: M. Vissani, 2018

% [V w St] = aEIF(t,I,C,gL,EL,Vt,Vp,Vr,Dt,tauw,a,b,initV,initw,pflag)
%
% t      - time vector in sec. (deltaT might influence the results; smaller values earlier spikes; 0.0005)
% I      - injected current (in nA); 1. dim: time course; 2. dim: different neurons
% C      - membrane conductance (in nF)
% gL     - leak conductance (in nS)
% EL     - leak reversal (in mV), e.g., -70
% Vt     - threshold potential for spike (in mV)
% Vp     - peak potential at spike (in mV), e.g. 20
% Vr     - reset potential after spike (in mV), e.g. Vr = EL; Vr > Vt -> bursting
% Dt     - rise slope factor (in mV); sharpness of spike
% tauw   - adaptation time constant (~Ca-activated K current inactivation; in ms)
% a      - 'sub-threshold' adaptation conductance (in nS); large a = subthreshold oscillations
% b      - 'sra' current increment (in nA); large b = strong spike-frequency adaptation
% initV - initial value of membrane potential V (in mV; default = EL)
% initw - initial value of adaptation current w (in nA; default = 0)
% pflag - plotting 1 or 0
%
% V  - membrane potential (in mV)
% w  - adaptation current (in nA)
% St - spike times
% convert to SI units
C      = par.C*1e-9;
gL     = par.gL*1e-9;
EL     = par.EL*1e-3;
Vt     = par.Vt*1e-3;
Vp     = par.Vp*1e-3;
Vr     = par.Vr*1e-3;
Dt     = par.Dt*1e-3;
a      = par.a*1e-9;
b      = par.b*1e-9;
tauw   = par.tauw*1e-3;
initV = par.initV*1e-3;
```

```
initw = par.initw*1e-9;
abs_T = par.abs_T*1e-3;
EE = par.EE*1e-3;
tau_syn = par.tau_syn*1e-3;
init_syn = par.init_syn;
g_ext = par.g_ext*1e-9;
% time step
dt    = t(2)-t(1);
% get number of neurons
nNeurons = par.N;
% set flag and counters to handle refractory mechanism
t_ref = zeros(1,nNeurons);
in_abs_ref= false(1,nNeurons);
% initialize state variables
[V, w, Sb, syn,I_ext ] = deal(zeros([length(t) nNeurons]));
V(1,:)   = initV;  % membrane potential vector
w(1,:)   = initw;  % adaptation current
syn(1,:) = init_syn; % external synapse
for ii = 1 : length(t)-1
    % if is in refractory period --- > mantain membrane voltage and
    % increase time after last spike or reduce refractory time
    V(ii+1,in_abs_ref) = Vr;
    t_ref(in_abs_ref) = t_ref(in_abs_ref) + dt;
  % update membrane potential(only if is not refracted)
    % launch this line for no adaptation
%   V(ii+1,~in_abs_ref) = V(ii,~in_abs_ref) + dt/C*(-gL*(V(ii,~in_abs_ref)-EL) +  gL*Dt*exp
    ((V(ii,~in_abs_ref)-Vt)/Dt)  -      g_ext*syn(ii,~in_abs_ref).*(V(ii,~in_abs_ref)-EE));
    % launch this line for adaptation
    V(ii+1,~in_abs_ref) = V(ii,~in_abs_ref) + dt/C*(-gL*(V(ii,~in_abs_ref)-EL) + gL*Dt*exp((
        V(ii,~in_abs_ref)-Vt)/Dt) - w(ii,~in_abs_ref) - g_ext*syn(ii,~in_abs_ref).*(V(ii,~in
        _abs_ref)-EE));
    % launch this line for adaptation and noise
    %V(ii+1,~in_abs_ref) = V(ii,~in_abs_ref) + dt/C*(-gL*(V(ii,~in_abs_ref)-EL) + gL*Dt*exp
        ((V(ii,~in_abs_ref)-Vt)/Dt) - w(ii,~in_abs_ref) - g_ext*syn(ii,~in_abs_ref).*(V(ii,~
        in_abs_ref)-EE) + sqrt(2*10^(-19))*randn());
    % check if the neuron will be outside its refractory time in the next
    % step
    in_abs_ref(t_ref >= abs_T) = false;
    t_ref(t_ref >= abs_T) = 0;
  % check for spikes
  ixf = V(ii+1,:) > Vp;
  % update adaptation conductance (update always!)
  w(ii+1,:) = w(ii,:) + dt/tauw*(a*(V(ii,:)-EL) - w(ii,:)) + ixf*b;
    % update external conductance (update always!)
    syn(ii+1,:) = syn(ii,:)*(1 - dt/tau_syn) + I(ii+1,:);
    % set spike and reset membrane potential
  V(ii,ixf)   = Vp;
    in_abs_ref(ixf) = true;
    V(ii+1,ixf)   = Vr;
    % save binary spikes train format
  Sb(ii,ixf)  = 1; % spike bits
end
I_ext = -g_ext*syn.*(V-EE);
% get spike times
if nNeurons == 1
    St = find(Sb)*dt;
else
    St = cell([nNeurons 1]);
    for ii = 1: nNeurons
        St{ii} = find(Sb(:,ii))*dt;
    end
end
% convert back to original units
V = V*1e3;
w = w*1e9;
I_ext = I_ext*1e9;
end
```

Synaptic Noise

To make more realistic the simulation, we will inject some noise in the neuron. The main component of the noise experienced by a neuron originates in the myriad of synapses made by other cells onto it. Every spike arriving at this synapse contributes a random amount of charge to the cell because switches that convert spikes into the release of fixed packets of neurotransmitter at synaptic boutons are not deterministic. In the limit where the strength of the weights of these synapses goes to zero and the frequency of incoming spikes goes to infinity, the sum of delta functions becomes Gaussian white noise. This is known as the diffusion limit of synaptic input. In this limit, synaptic noise can essentially be approximated by additive Gaussian white noise $\xi(t)$ [7, 8]. The resulting system becomes:

$$\underbrace{C\frac{\mathrm{d}V(t)}{\mathrm{d}t}}_{\text{Membrane}} + \underbrace{g_L(V(t)-E_L)}_{\text{Leakage}} - \underbrace{g_L D_T e^{\frac{V(t)-V_T}{D_T}}}_{\text{Exponential}} + \underbrace{w(t)}_{\text{Adaptation}} + \underbrace{g_{Ext}s(t)(V(t)-E_E)}_{\text{Input}} + \underbrace{\sqrt{(2\tau_N)}\xi(t)}_{\text{Noise}} = 0 \tag{9}$$

which is also known as an Ornstein-Uhlenbeck (OU) process with correlation time τ_N [9]. The numerical implementation is not trivial because of the time scaling of the noise. Suppose we just had the differential equation:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \xi \tag{10}$$

To solve this numerically we could compute:

$$x(t+h) = x(t) + \xi_1 \tag{11}$$

where ξ_1 is a normally distributed random number with mean 0 and standard deviation 1. However, what happens if we change the time step? Suppose we used a value of $\frac{h}{2}$ instead of h. Now, we compute:

$$x(t+h) = x(t+\frac{h}{2}) + \xi_1 = x(t) + \xi_2 + \xi_1 \tag{12}$$

The mean value of $x(t+h)$ is 0 in both cases, but the standard deviations are different. The first method $x(t+h) = x(t) + \xi_1$ gives $x(t+h)$ a standard deviation of 1, whereas the second method $x(t+h) = x(t+\frac{h}{2}) + \xi_1 = x(t) + \xi_2 + \xi_1$ gives $x(t+h)$ a variance of $1+1=2$ and therefore a standard deviation of $\sqrt{2}$.
To solve this problem, we use the Euler-Maruyama method, a generalization of the Euler method for Stochastic Differential Equations [10]:

$$x(t+h) = x(t) + \sqrt{h}\xi_1 \tag{13}$$

which makes the mean and standard deviation of the value at time t independent of h. For this to make sense dimensionally, ξ must have units of $[s^{-\frac{1}{2}}]$.

References

[1] Romain Brette and Wulfram Gerstner. "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity". In: *Journal of Neurophysiology* (2005). ISSN: 00223077. DOI: 10.1152/jn.00686.2005.

[2] Peter Dayan and L F Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. 2001.

[3] Richard Naud et al. "Firing patterns in the adaptive exponential integrate-and-fire model". In: *Biological Cybernetics* 99.4 (Nov. 2008), pp. 335–347. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/s00422-008-0264-7. URL: http://link.springer.com/10.1007/s00422-008-0264-7 (visited on 05/06/2020).

[4] Rimjhim Tomar. "Review: Methods of firing rate estimation". In: *Biosystems* 183 (Sept. 2019), p. 103980. ISSN: 03032647. DOI: 10.1016/j.biosystems.2019.103980. URL: https://linkinghub.elsevier.com/retrieve/pii/S0303264719301492 (visited on 05/06/2020).

[5] John J. Gart. "The Poisson Distribution: The Theory and Application of Some Conditional Tests". In: *A Modern Course on Statistical Distributions in Scientific Work*. Ed. by G. P. Patil, S. Kotz, and J. K. Ord. Dordrecht: Springer Netherlands, 1975, pp. 125–140. ISBN: 978-94-010-1845-6.

[6] B N Biswas et al. "A DISCUSSION ON EULER METHOD: A REVIEW". In: *Electronic Journal of Mathematical Analysis and Applications* 1 (June 2013), pp. 294–317.

[7] Petr Lansky and Vera Lanska. "Diffusion approximation of the neuronal model with synaptic reversal potentials". In: *Biological cybernetics* 56 (Feb. 1987), pp. 19–26. DOI: 10.1007/BF00333064.

[8] R.M. Capocelli and L.M. Ricciardi. "Diffusion approximation and first passage time problem for a model neuron". In: *Kybernetik* 8 (June 1971). DOI: 10.1007/BF00288750.

[9] Henry Clavering Tuckwell and Henry C. Tuckwell. *Nonlinear and stochastic theories*. Digitally printed 1. paperback version. Introduction to theoretical neurobiology Vol. 2. OCLC: 934968414. Cambridge: Cambridge Univ. Press, 2005. 265 pp. ISBN: 978-0-521-01932-3 978-0-521-35217-8.

[10] Sunday Kayode, Akeem Ganiyu, and Adegoke Ajiboye. "On One-Step Method of Euler-Maruyama Type for Solution of Stochastic Differential Equations Using Varying Step-sizes". In: *OALib* 03 (Jan. 2016), pp. 1–15. DOI: 10.4236/oalib.1102247.

Excercise: Integrate and Fire model

Exercise 1. Run the simulation with the default parameters.

(a) What is the behavior of the neuron with an external $\lambda = 300Hz$ stimulus?

- ◯ Silent period.
- ◯ Tonic firing pattern with an average $FR = 150Hz$.
- ◯ Burst-like firing pattern with an average $FR = 150Hz$.
- ◯ Sporadic firing pattern.

(b) The stereotyped shape of an action potential is characterized by three particular events: a snap increase in the voltage $V \simeq -41mV$, a peak $V \simeq 20mV$ and a reversion to $V \simeq -47mV$. What are the neural and computational mechanisms that underlie them?

- ◯ K^+ positive feedback (exponential term), K^+ influx is null (threshold mechanism) and K^+ rest (reset mechanism).
- ◯ Na^+ positive feedback (exponential term), Na^+ influx is null (threshold mechanism) and Na^+ rest (reset mechanism).
- ◯ Na^+ influx is null (exponential term), Na^+ positive feedback (threshold mechanism) and Na^+ rest (reset mechanism).

(c) What happens after a spike is triggered? On which neural feature this mechanism puts a theoretical boundary? Upper or lower?

- ◯ There is a flat trend because of the relative refractory time. It puts a upper boundary to the firing rate.
- ◯ There is a flat trend because of the absolute refractory time. It puts an upper boundary to the firing rate.
- ◯ There is a flat trend because of the absolute refractory time. It puts a lower boundary to the firing rate.

(d) What happens to the w variable?

- ◯ It decreases instantaneously of an amount b after each spike and it increases exponentially with a time constant determined by a between two spikes.
- ◯ It increases instantaneously of an amount b after each spike and it decreases exponentially with a time constant determined by a between two spikes.
- ◯ It decreases instantaneously of an amount a after each spike and it increases exponentially with a time constant determined by b between two spikes.
- ◯ It increases instantaneously of an amount a after each spike and it decreases exponentially with a time constant determined by b between two spikes.

(e) Which term in the membrane voltage differential equation determines the shape of the action potential once the voltage reached the threshold?

- ◯ The leakage term.
- ◯ The exponential term.
- ◯ The adaptation term.
- ◯ The external stimulus term.

(f) What is the effect of the adaptation variable?

- ◯ It acts as a positive current to the neuron proportional to the firing rate (positive feedback).
- ◯ It acts as a positive current to the neuron independent of the firing rate (constant positive feedback).
- ◯ It acts as a negative current to the neuron proportional to the firing rate (negative feedback).
- ◯ It acts as a negative current to the neuron independent of the firing rate (constant negative feedback).

(g) What happens when the stimulus vanishes?

- ◯ A strong hyperpolarization due to the residual effect of the adaptation w variable
- ◯ A strong depolarization due to the residual effect of the adaptation w variable

Exercise 2. Explore the parameters in the model.

(a) Increase the duration of the stimulus untill $1.5s$. What happens if $a = b = 0$? How can you obtain the same effect? What happens if $a = 0 \wedge b \neq 0$ or $a \neq 0 \wedge b = 0$? *Hint:* compare firing rates and give a look to Eqs 1-2.

(b) What is approximately the minimum frequency of the external stimulus able to elicit at least 1 spike in the neuron? Play with the λ parameter and once you identified it keep in mind it as λ_R.

(c) Use λ_R and change the susceptibility of the neuron varying V_T. What happens if $V_T = -30mV$ or $V_T = -55mV$? Does V_p influence the neuronal dynamic?

(d) Use $\lambda < \lambda_R$ and inspect the shape of Excitatory Postsynaptic Current (Input variable, EPSC) and Excitatory PostSynaptic Potential (Voltage, EPSP).

(e) What is the shape of the EPSC? What determines the velocity of the decay of the synapse? What is the shape of the EPSP? What determines the rise and the decays of the EPSP?

(f) Do you think that the membrane voltage is able to follow instantaneously the synaptic input?

- ◯ Yes, the neuron acts as a all-pass filter.
- ◯ No, the neuron acts as a high-pass filter with constant time $\tau = \frac{C}{g_{Ext}}$
- ◯ No, the neuron acts as a low-pass filter with constant time $\tau = \frac{C}{g_{Ext}}$

(g) Use $\lambda < \lambda_R$ and change the polarity of the input (i.e. inhibitory). How can you do it? *Hint: look at to the term* $(V - E_E)$. Inspect the shape of Inhibitory Postsynaptic Current (Input variable, IPSC) and Inhibitory PostSynaptic Potential (Voltage, IPSP).

(h) Use $\lambda < \lambda_R$ with an excitatory input. Try to increase the strength of the synapse changing the variable g_{Ext} until the neuron is able to fire with the same λ.

(i) Return to the original parameters set and use a high λ. Try to change the W_k of the kernel $K(t)$ of the FR. What happens to the FR esteem if you increase or decrease W_k?

Exercise 3. Synaptic temporal Summation.

Use $\lambda < \lambda_R$. Sometimes the synapse is able to trigger the neuron successfully even if it's weak because the temporal summation mechanism. Multiple close EPSCs can build up until the threshold of the voltage membrane is crossed.

(a) Run the simulation until you encounter into a temporal summation phenomenon. Is it successful? If yes or no, why?

(b) You can facilitate the temporal summation effect changing the constant time of the synapse τ_s. What do you expect if you increase τ_s?

Insight: You can interpret "increasing the frequency input λ" as another way to increase the likelihood to occur a temporal summation effect. More spikes you have and more is the probability that multiple spikes are enough close to build up together.

Exercise 4. Noise in the model. Try to inject OU noise with correlation time τ_N in the model using the Euler-Maruyama method.

(a) Use $\lambda < \lambda_R$ and $\tau_N = 10^{-17}$ and see what happens.

(b) Try different level of noise and see what happens if the noise is too strong.