MASTER'S DEGREE IN PHYSICS
Academic Year 2020-2021

BIOLOGICAL PHYSICS

Student: Giorgio Palermo
Student ID: 1238258
Date: November 2, 2020

DERIVATION OF CELL PARAMETERS

In this report I will describe how bla bla

1 Suca

The cell membrane is the membrane that surrounds and encloses the cytoplasm and the nucleus of a living cell. It is formed by a lipid bilayer and includes several kinds of membrane proteins, which perform important physiological functions such as signal transmission, ion transport and cell adhesion. One important parameter which is strictly related to the ion transport properties of the cell membrane is the *membrane potential* of the cell, which is defined as the potential difference between the intracellular and extracellular potential:

$$V_m = V_{in} - V_{ex}.$$

Transport of ions across the proteins situated inside the membrane causes the modification of both the external and internal concentrations of ions and this leads to a modification of the membrane potential. In laboratory environment the external potential can often be set to a constant, so membrane potential variations reflect changes in the internal concentration of ions. In this situation and in absence of external stimuli, the membrane potential is referred to as *resting potential*.

Patch clamp is an experimental technique used in electrophysiology to study ionic currents in individual isolated living cells. It consists in fixing the potential difference in a small area of the cell membrane or in the whole cell and then look to current variations in order to study for example ionic channels response to potential variations or more complex cell processes. It can be used on cell cultures, isolated cells or even on brain slices.

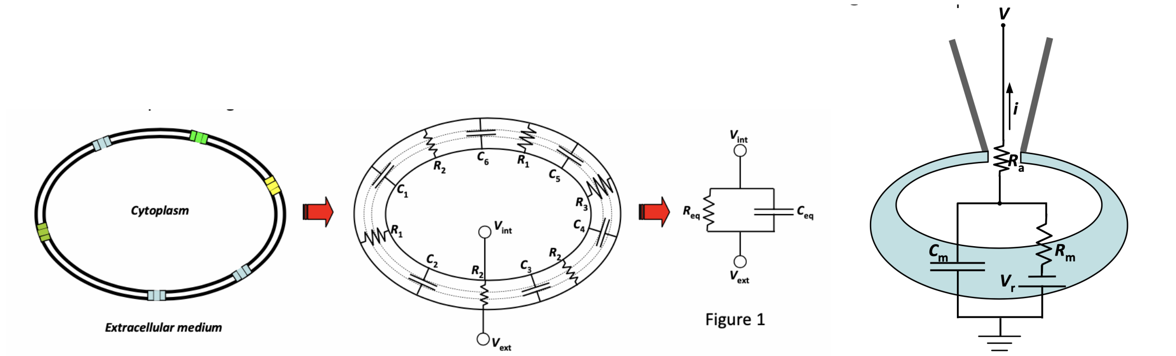
The measurement is performed using a single microelectrode made by a glass micropipette. The point of this micropipette presents a hole with diameter of approximately 1 μm and resistance (*access resistance*) of 1 to 10 MOhm. This extremity is made to adhere to a small area of cell membrane (patch), thus isolating the ion channels. At this point it is possible to manipulate the ion channels altering the chemical composition of the fluid placed inside the pipette or the electrical properties of the membrane.

Figure 1: Typical current record from a single channel patch clamp experiment.

2 Minchie

The simulation performed in this exercise is aimed to discover the response of a cell in a whole cell patch clamp configuration to a square wave electrical stimulus.

In order to get meaningful predictions on the behavior of the cell one must summarize the electrical properties of the system in an electrical scheme, namely the one in figure 2b. As a first approximation, one can consider the cell membrane as made of a lipid bilayer and ion channels, which can be electrically



(a) (b)

Figure 2: **(a)** Derivation of the electrical scheme for a cell; **(b)** Schematic representation of the electrical properties of a cell

schematized by capacitors and resistances. The fact that both these components experience the same potentials at the two sides, leads us to the conclusion that they must be connected in parallel. We will indicate the total capacity and resistance of the cell membrane with C_m and R_m. As in this exercise we are considering whole cell patch clamp configuration, access resistance (R_a) of the pipette must be considered as well; the voltage applied is controlled by the experimenter by setting the pipette potential: we will indicate this value with V.

In the following passages we will derivate the equations that link the electrical parameters of the simulation C_m, R_m, R_a and V to the current i flowing through the cell.

The system is described by the first order linear differential equation derived from Kirchoff laws:

$$\tau \frac{\mathrm{d}V_m}{\mathrm{d}t} = -V_m + V \frac{R_m}{R_m + R_a} \quad \text{with} \quad \tau = \frac{R_a R_m}{R_a + R_m} C_m, \tag{1}$$

where V_m is the membrane potential as defined above. The current flowing through the membrane in the small time interval $[t, t + \mathrm{d}t]$ can be computed as

$$i(t + \mathrm{d}t) = \frac{V_m(t)}{R_m} + C_m \frac{\mathrm{d}V_m}{\mathrm{d}t}(t) \tag{2}$$

A numerical solution for the behavior of the current signal in time is obtained with time discretization and the implementation of (2) and (1) as update rules at each step. In pseudo-code using j as time index it results in:

$$\begin{cases} \frac{\mathrm{d}V_m}{\mathrm{d}t}(j) = -V_m(j) + V_{in}(j) \frac{R_m}{\tau(R_m + R_a)} \\ i(j) = \frac{Vm(j)}{R_m} + C_m \frac{\mathrm{d}V_m}{\mathrm{d}t}(j) \\ Vm(j+1) = V_m(j) + \mathrm{d}V_m \, \mathrm{d}t \end{cases} \tag{3}$$

Note that in this expression we set V_{in} as a time dependent quantity: this is motivated by the fact that we want to simulate the behavior of the pipette-cell system for an input voltage that has the shape of a square wave.

The result of such numeric process is the numerical estimation of the behavior of the current as a function of time. This output, as depicted with a red line in figure 3a, is a smooth function of time. However in real laboratory situations noise come into play, modifying the ideal shape of the signal from a smooth curve to a disturbed one. Therefore, in order to produce a more realistic simulation, noise has been added to the signal (blue line in figure 3a). The distribution of the noise of the signal is non trivial and could in principle depend on many factors, such as cell parameters like membrane resistance or capacitance, ambient parameters like temperature or on factors related to the experimental apparatus itself. In this work I chose to assign Gaussian distribution to noise for simplicity; I set the amplitude of this noise in order to make the signal resemble the example shown in class during the discussion of Patch Clamp experiments.

In a laboratory situation we would be interested in retrieving the experimental parameters from data, namely $\tilde{R}_a, \tilde{R}_m$ and $\tilde{C}_m$. To perform this task we are helped by the analytical solution for current of the system described above in the case of a square wave of amplitude V:

$$i(t) = \frac{V\left(1 + \frac{R_a}{R_m}\right)}{R_a + R_m} e^{-\frac{t-t_0}{\tau}} \tag{4}$$

which is equal to:

$$i(t) = \begin{cases} \frac{V}{R_a} & \text{if } t = t_0 \\ \frac{V}{R_a + R_m} & \text{if } t >> \tau. \end{cases} \tag{5}$$

Starting from this expression and calling $i(t = t_0) = i_{pk}$ and $i(t >> \tau) = i_\infty$ one can use these two current values to retrieve:

$$\tilde{R}_a = \frac{V}{i_{pk}} \quad \text{and} \quad \tilde{R}_m = \frac{V}{i_\infty} - R_a \tag{6}$$

Finally, the estimation of C_m is done using (1) and the experimental value of τ :

$$\tilde{C}_m = \tilde{\tau}\left(\frac{1}{\tilde{R}_a} + \frac{1}{\tilde{R}_m}\right). \tag{7}$$

Similar equations hold also when the cell presents a resting potential, namely if we call it V_r the analytical solution becomes:

$$i(t) = i_\infty + \frac{V \frac{R_a}{R_m} e^{-\frac{t-t_0}{\tau}}}{R_a + R_m} \quad \text{with} \quad i_\infty = \frac{V - V_r}{R_a + R_m} \tag{8}$$

and the equations for the experimental parameters become:

$$\hat{R}_a = \left[\frac{i_{pk} - i_\infty}{V} + \frac{i_\infty}{V - V_r}\right]^{-1} \tag{9}$$

$$\hat{R}_m = \frac{V - V_r}{i_\infty} - \hat{R}_a \tag{10}$$

$$\hat{C}_m = \tau\left(\frac{1}{\hat{R}_a} + \frac{1}{\hat{R}_m}\right) \tag{11}$$

The final part of the simulation involved, as in real experiments, the presence of a 4-pole low pass Bessel filter aimed to "clean" the signal from high-frequency noise. As will be discussed later, the presence of this component effectively reduce the HF signal noise over the threshold frequency, however a side effect is introduced: signal peaks are shifted in time and lowered in magnitude. This behavior is particularly bad for our estimation, since the derivation of the access resistance is performed starting from the peak value of the current (see (6)) that is no more accurately reported by the experimental setup. To overcome this problem, another method can be used to esteem the experimental value of the access resistance $\tilde{R}_a$. Starting from equation (1) and its solution for voltage, namely

$$V_m(t) = \frac{V R_m}{R_a + R_m}\left(1 - e^{-t/\tau}\right) \tag{12}$$

one can express the current flowing through R_a as the sum of a regime current and a transient current

$$i_{R_a} = i_{C_m} + i_{R_m} = i_{tr} + i_\infty \quad \text{with} \quad i_{tr} = Ae^{-t/\tau}, \quad A \text{ constant.} \tag{13}$$

The total charge transfered from transient current is given by:

$$Q_{tr} = \int_0^\infty Ae^{-t/\tau}\, \mathrm{d}t = \frac{i_{tr}}{\tau} \tag{14}$$

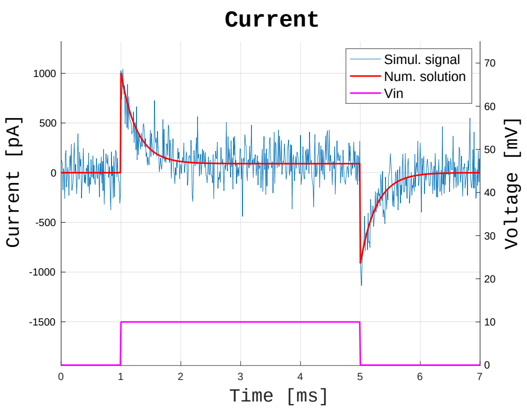
therefore, substituting in (13) we get

$$i_{R_a} = \frac{Q_{tr}}{\tau} + i_\infty \tag{15}$$

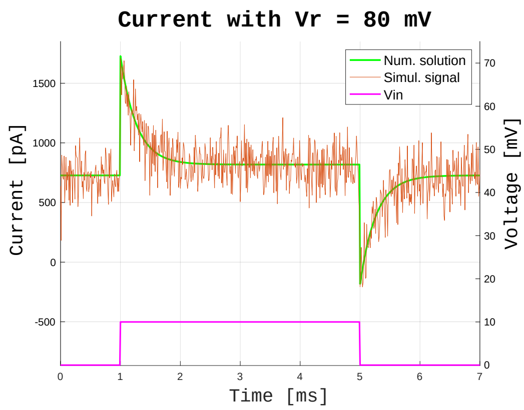
and finally

$$\tilde{R}_a = \frac{V}{i_{R_a}} = \frac{\tau V}{Q_{tr} + \tau i_\infty}. \tag{16}$$

This procedure can be implemented in the offline digital analysis by numerically integrating the signal to get the transferred charge and then retrieve the experimental value for the access resistance.



(a)



(b)

Figure 3: **(a):** Current signal; **(b):** current signal with 80 mV resting potential; below each graph the input voltage is represented in magenta

3 Results, finally! :-)

In this report I simulated the response of a cell-pipette system characterized by the parameters: R_a = 100 MΩ; R_m = 10 MΩ; and C_m = 30 pF. The simulation lasts 7 ms and the sampling rate (spacing of the time grid) has been chosen in accordance to a realistic value that one could achieve from a real experiment, namely 100 kHz.

Figure 3a and 3b show the computed simulated current in the case of no resting potential and resting potential V_r = 80 mV; in both figures the input square wave is drawn in magenta. Red and green solid lines represent the numerical solution of the differential equation describing the system; adding Gaussian noise as described before I obtained the wavy line, which is a more realistic representation of the signal that one could retrieve from an experiment.
In order to estimate R_a and R_m the values of i_∞ and i_{pk} has been retrieved from data. The value of i_∞ has been estimated averaging the current values for times larger than five times the time constant of the exponential decay, while the value of i_{pk} has been established averaging the first five current samples after the peak. This method was used in order to reduce the dependence of the final values of R_a and R_m from the stochastic error on the current value given by the noise.
To retrieve the time constant of the exponential decay, an exponential fit has been performed, as can be seen in figure 4. The value of τ retrieved from this procedure has been inserted in equation (7) to evaluate C_m. Table 1 gathers the values of the experimental parameters for the case with no resting potential and the case in which V_r = 80 mV.

The last part of the exercise was related to attenuation of the signal noise using a 4-pole, low pass Bessel filter. In laboratory situation as well as in this simulation, this electronic component acts on the signal attenuating all the frequencies higher than a threshold value (hence the name: "low pass"). In this exercise the filter has been simulated with a built-in Matlab function, `belsself(n,omega)`, which is able to reproduce a n-pole Bessel filter with cutoff frequency ω. To quantify the modification that the filter applies to the signal, the filtering procedure and the analysis has been repeated for different frequencies, namely six values between 1 and 10 kHz, that I chose to be log-spaced. For each cutoff frequency the signal has been filtered and then all the analysis described above was repeated, namely computing $\tilde{R}_a, \tilde{R}_m$ fitting the exponential decay and computing $\tilde{C}_m$. The filtered signals are plotted in figure 6; from this figure we can see that the lower the threshold frequency, the minor the noise contribution to the signal. For the purpose of comparison, all the filtered signals at different cutoff frequencies are plotted together in figure 5.

4 Discussion

The current simulation has been performed starting from the differential equation describing the system; this simulation successfully reproduced the behavior described in the lecture notes. After the noise was added, experimental parameters have been retrieved from the sample data. It is due to the

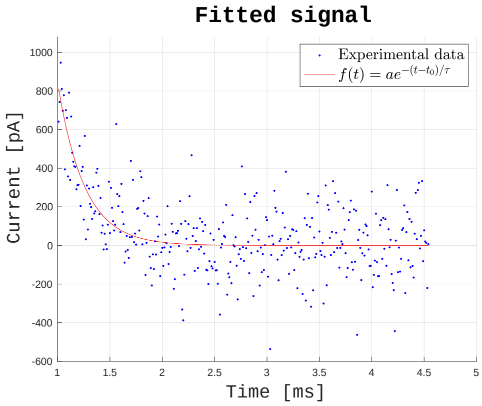


Figure 4: Portion of the signal fitted to retrieve τ

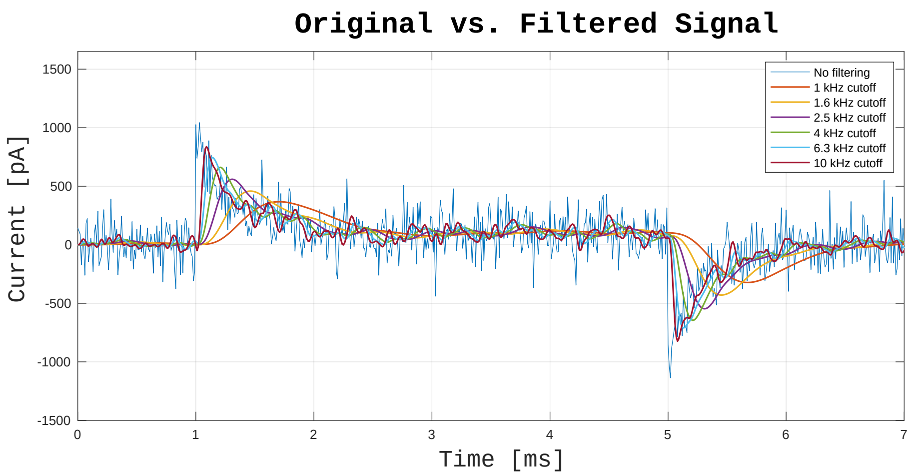


Figure 5: Comparison between original noisy signal and filtered signals at various cutoff frequencies

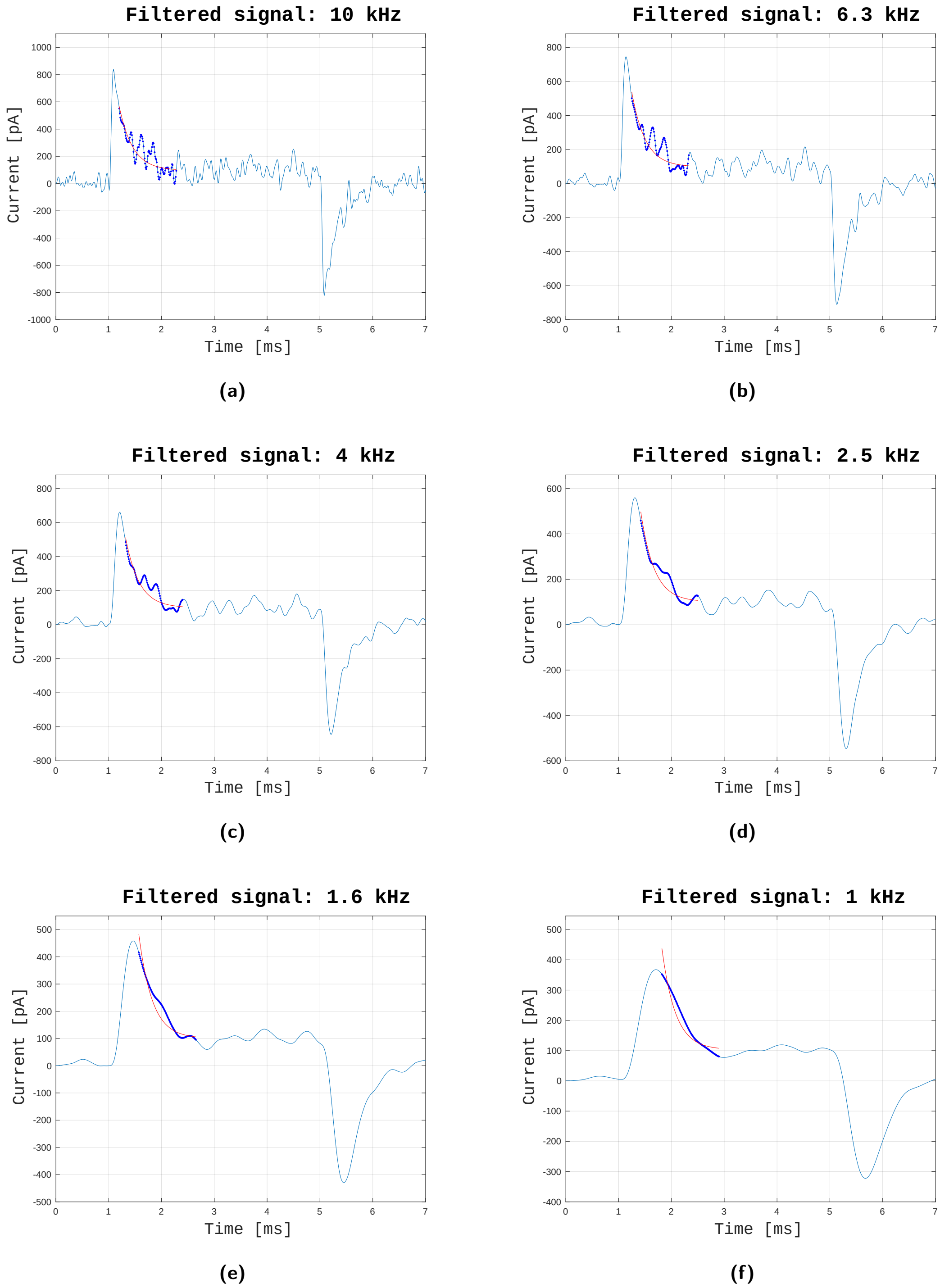


Figure 6: Filtered signals at various cutoff frequencies, namely: 10, 6.3, 4, 3, 2.5, 1.6 and 1 kHz

Table 1: Experimental parameters retrieved from simulation, unfiltered signal

$\boldsymbol{V_r}$ [mV]	$\tilde{\boldsymbol{R}}_{\boldsymbol{a}}$ [MΩ]	$\tilde{\boldsymbol{R}}_{\boldsymbol{m}}$ [MΩ]	$\tilde{\boldsymbol{C}}_{\boldsymbol{m}}$ [pF]
0	11.1 ± 0.7	107.4 ± 11	26 ± 5
80	12.6 ± 0.5	96 ± 11	22 ± 5

Table 2: Experimental parameters retrieved from simulation, filtered signals

Cutoff [Hz]	$\tilde{\boldsymbol{R}}_{\boldsymbol{a}}$ [MΩ]	$\tilde{\boldsymbol{R}}_{\boldsymbol{m}}$ [MΩ]	$\tilde{\boldsymbol{C}}_{\boldsymbol{m}}$ [pF]
1000	21.5 ± 0.7	83 ± 1	14.9 ± 0.6
1585	14 ± 1	90 ± 2	22 ± 3
2512	12 ± 1	91 ± 2	24 ± 4
3981	12 ± 1	91 ± 2	25 ± 4
6310	11 ± 2	93 ± 3	25 ± 5
10000	11 ± 2	95 ± 4	25 ± 5

presence of the noise the fact that these esteems do report a considerable error, of the order of 5-10% of the real value. Although it might seem a poor result, we were able to estimate the uncertainty of a real measurement via simulation: it could be interesting to determine via simulation which is the kind of noise that mostly contribute to the uncertainty of experimental parameters.

The second simulation was performed introducing membrane resting potential. This led to an increase of the total current, both the peak value during the transient and the rest value. This fact hasn't led to visible improvement in the quality of the estimations. It is interesting to notice that the increase in the current value is about 750 pA (see figure 3b). Such an increase has not made the difference in our case, but in a situation of lower noise it could have led to a significantly better signal-to-noise ratio.

Comparing figure 3a with 6 we can see that the filtering procedure significantly increase the cleanness of the signal. However, there is a drawback: the transient peak of the signal is both lowered and delayed in time. A quick overview of this phenomenon is given in figure 5, where differently filtered signals are plotted against the original one. Looking at the parameters in table 2 we see that, increasing the cutoff frequency we get better and better estimations for all the experimental parameters with respect to the real values set at the beginning of the simulation. The uncertainties of these measurements become larger for higher cutoff frequencies: this is due to the fact that the signal is more scattered; on the other hand, the expected value is more accurate. This simulation allows us to understand the relation between filtering and loss: the more we filter the signal making it smooth and scarcely fluctuating, the more information we lose about the shape and the magnitude of the peak.

5 References

Hodgkin & Huxley, *A quantitative description of membrane current and its application to conduction and excitation in nerve*, The Journal of Physiology, 1952

Jackson M. B., *Molecular and cellular biophysics*, Cambridge University Press, 2006

H. Sontheimer, *Whole-Cell Patch-Clamp Recordings*, Neuromethods, Vol 26 Patch-Clamp Applications and Protocols, Humana Press, 1995

Lecture notes from the course *Biological Physics*

6 Appendix

This section contains the code developed to solve the exercise.

```
% Giorgio Palermo
% Padova, 12/18/2020
% Course: Biological Physics
% Exercise: Derivation of cell parameters
% ------------------ Problem parameters ------------------------
clear all
% Universal constants
kb=1.38e-23;
% Simulation parameters
sr= 1e5; % Hz, sampling rate of the virtual oscilloscope
dur= 7e-3; % seconds, duration of the simulation
tstart=1e-3; % limits on the simulation
tstop=5e-3;
Vin_amplitude = 10e-3; % Square wave amplitude
tstep=1/sr;
time=(0:tstep:dur)';
Nstep=size(time,1)-1;
noise_amp= 1.5e-10;
% Cell parameters
disp('Computing signal... ')
Ra= 1e7; %Ohm, access resistance
Rm=1e8; %Ohm, membrane resistance
Cm=3e-11; %F, membrane capacity
tau=Cm*(1/Rm + 1/Ra)^(-1);

% ------------------ No resting potential -----------------------
% Initializing vectors
Vin=zeros(Nstep+1,1);
Vm=zeros(Nstep+1,1);
I0=zeros(Nstep+1,1); % Current without noise
I1=I0; % Current with noise
Vin=step_fun(Vin,time,tstart,tstop,Vin_amplitude); %Square wave

%Print input voltage
f10=figure(10);
f10.Visible='off';
clf
pl1=plot(time*1e3,Vin*1e3,'Linewidth',1.5);
prop = {"Input voltage", "Time [ms]", "Voltage [mV]","m","y"};
SetPlot(get(gcf), prop)
printpdf(f10,'Voltage_input');

% Solve the differential equation
for i=1:1:Nstep
  dVm=(-Vm(i)+Vin(i)*Rm/(Rm+Ra))/tau;
  I0(i) = Vm(i)/Rm+Cm*dVm;
  Vm(i+1) = Vm(i)+dVm*tstep;
  I1(i)=I0(i)+noise_amp*randn;
end

% Plot input voltage and current result
f20 = figure(20);
f20.Visible='off';
clf
hold on
grid on
pl21=plot(time*1e3,I1*1e12,'DisplayName','Simul. signal');
pl2=plot(time*1e3,I0*1e12,'r-','Linewidth',1.5,'DisplayName','Num. solution');
prop = {"Current", "Time [ms]"," Current [pA]","m","y"};
SetPlot(get(gcf), prop)
ax=f20.CurrentAxes;
ax.YLim=enlarge(ax.YLim,0.07);
ax.YLim=pushup(ax.YLim,0.27);
hold on
yyaxis right
prop = {"Current", "Time [ms]"," Voltage [mV]","m","y"};
SetPlot(get(gcf), prop)
ax=f20.CurrentAxes;
ax.YLim = [-.10 75];
ax.YColor = [0 0 0];
```

```
pl1=plot(time*1e3,Vin*1e3,'-m','Linewidth',1.5,'DisplayName','Vin');
legend('FontSize', 13);
disp('Saving plot... ')
printpdf(f20,'Current')

%Ra, Rm estimation:
peak1=PeakEstim(time,I1,tstart,5);
Ra1=Vin_amplitude/peak1;
I1_infty_start = ceil((tstart+5*tau)/tstep); % I_infty estimation
I1_infty_end = ceil(tstop/tstep)-10;
I1_infty=mean(I1(I1_infty_start:I1_infty_end));
Rm1=Vin_amplitude/I1_infty - Ra1;

%Cm estimation via curve fitting
x= time(time>tstart & time<tstart+13*tau);  %Select right time interval
y=I1(time>tstart & time<tstart+13*tau) - I1_infty; %subtract baseline
fitres=fit(x,y,'exp1','startpoint',[1e-9,-1/0.2545e-3]); %fit specifying initial
    parameters
tau1=-1/fitres.b;
Cm1=tau1*(1/Ra1 +1/Rm1);
y1=fitres(x);

%Plotting fitted data
f30 = figure(20);
f30.Visible='off';
clf
grid on
hold on
pl31=plot(x*1e3,y*1e12,'b.');
pl3=plot(x*1e3,y1*1e12,'r-');
prop = {"Fitted signal", "Time [ms]"," Current [pA]","m","y"};
SetPlot(get(gcf), prop)
legend('Experimental data','$f(t) = a e^{-(t-t_0)/\tau}$','Interpreter','latex','
    FontSize',15);
disp('Saving plot... ')
printpdf(f30,'Current_fitted')

% Errors
sigpt=noise_amp;
peak1_sig=sigpt/sqrt(5);
Ra1_sig=Ra*peak1_sig/peak1;
I1_infty_sig=sigpt/sqrt(size(I1(I1_infty_start:I1_infty_end),1));
Rm1_sig=sqrt(Rm^2*(I1_infty_sig/I1_infty)^2 + Ra1_sig^2);
bounds=confint(fitres);
bounds=-(1./bounds);
bounds(:,1)=[];
tau1_sig=(bounds(2)-bounds(1))/2;
R=(1/Ra1 + 1/Rm1);
R_sig=sqrt((Ra1_sig/Ra^2)^2 + (Rm1_sig/Rm1^2)^2);
Cm1_sig=Cm1*sqrt((tau1_sig/tau1)^2 + (R_sig/R)^2);

% ------------------ Resting potential -------------------
disp('Computing signal with resting potential... ')
Vrest=-0.08; %mV, resting potential
I0r=I0-I0;
I1r=I0r;
Vmr=zeros(Nstep+1,1);
Vmr(1)=Vrest*Ra/(Rm+Ra);
% Solving differential equation
for i=1:1:Nstep+1
  Vc=(Rm*Vin(i)+Ra*Vrest)/(Ra+Rm);
  dVmr=-1/tau*(Vmr(i)-Vc);
  I0r(i) = ((Vmr(i)-Vrest)/Rm + Cm*dVmr);
  I1r(i)=I0r(i)+noise_amp*randn;
  Vmr(i+1) = Vmr(i)+dVmr*tstep;
end

%Ra, Rm estimation:
peak1=PeakEstim(time,I1r,tstart,5);
I1r_infty_start = ceil((tstart+7*tau)/tstep); %I1_infty estimation
I1r_infty_end = ceil(tstop/tstep)-10;
I1r_infty=mean(I1r(I1r_infty_start:I1r_infty_end));
Ra1r = ((peak1-I1r_infty)/Vin_amplitude + I1r_infty/(Vin_amplitude-Vrest))^-1;
```

```
Rm1r=(Vin_amplitude-Vrest)/I1r_infty - Ra1r;
tau1r=0;
Cm1r=0;

%Cm estimation via curve fitting
xr= time(time>tstart & time<tstart+13*tau); %Select right time interval
yr=I1r(time>tstart & time<tstart+13*tau) - I1r_infty; %subtract baseline
fitres1=fit(xr,yr,'exp1','startpoint',[1e-9,-1/0.2545e-3]); %fit specifying initial
    parameters
tau1r=-1/fitres.b;
Cm1r=tau1r*(1/Ra1r +1/Rm1r);
y1r=fitres(xr)+I1r_infty;

% Errors
sigpt=noise_amp;
peak1_sig=sigpt/sqrt(5);
temp=sqrt(Vin_amplitude^-2*(peak1_sig^2 + I1_infty_sig^2) + (I1_infty_sig^2/(
    Vin_amplitude-Vrest))^2);
Ra1r_sig = Ra1r^2*temp;
I1r_infty_sig=sigpt/sqrt(size(I1r(I1r_infty_start:I1r_infty_end),1));
Rm1r_sig=sqrt(Rm1r^2*(I1r_infty_sig/I1r_infty)^2 + Ra1r_sig^2);
bounds=confint(fitres1);
bounds=-(1./bounds);
bounds(:,1)=[];
tau1r_sig=(bounds(2)-bounds(1))/2;
R=(1/Ra1r + 1/Rm1r);
R_sig=sqrt((Ra1r_sig/Ra1r^2)^2 + (Rm1r_sig/Rm1r^2)^2);
Cm1r_sig=Cm1r*sqrt((tau1r_sig/tau1r)^2 + (R_sig/R)^2);

% Plot input voltage and current result
f40=figure(40);
f40.Visible='off';
clf
hold on
grid on
pl2=plot(time*1e3,I0r*1e12,'g-','Linewidth',1.7,'DisplayName','Num. solution');
pl21=plot(time*1e3,I1r*1e12,'DisplayName','Simul. signal');
tit="Current with Vr = " + string(-Vrest*1e3) +" mV";
prop = {tit, "Time [ms]", "Current [pA]","m","y"};
SetPlot(get(gcf), prop)
ax=f40.CurrentAxes;
ax.YLim=enlarge(ax.YLim,0.07)
ax.YLim=pushup(ax.YLim,0.27);
hold on
yyaxis right
prop = {tit, "Time [ms]"," Voltage [mV]","m","y"};
SetPlot(get(gcf), prop);
ax=f40.CurrentAxes;
ax.YLim = [-.10 75];
ax.YColor = [0 0 0];
pl1=plot(time*1e3,Vin*1e3,'-m','Linewidth',1.5,'DisplayName','Vin');
legend('FontSize', 13);
disp('Saving plot... ')
printpdf(f40,'Current_r')

% Plot fitted data
f50=figure(50);
f50.Visible='off';
clf
hold on
grid on
pl5=plot(xr*1e3,y1r*1e12,'r-',xr*1e3,yr*1e12+I1r_infty,'b.');
prop = {tit + "fit", "Time [ms]", "Current [pA]","m","y"};
SetPlot(get(gcf), prop)
disp('Saving plot... ')
printpdf(f50,'Current_r_fitted')

% Display some results on screen
disp('-------------- Caso base ----------------------------------')
varnames={'I1_infty [pA]','Ra1 [MOhm]','Rm1 [MOhm]','tau1 [ms]','Cm1 [pf]'};
T=table([I1_infty,I1_infty_sig]*1e12, [Ra1,Ra1_sig]*1e-6, [Rm1,Rm1_sig]*1e-6,[tau1,
    tau1_sig]*1e3,[Cm1,Cm1_sig]*1e12,'VariableNames',varnames);
disp(T);
```

```
disp('-------------- Caso resting potential --------------------')
varnames={'I1r_infty [pA]','Ra1r [MOhm]','Rm1r [MOhm]','tau1r [ms]','Cm1r [pf]'};
Tr=table([I1r_infty,I1r_infty_sig]*1e12, [Ra1r,Ra1r_sig]*1e-6, [Rm1r,Rm1_sig]*1e-6,[
    tau1r,tau1r_sig]*1e3,[Cm1r,Cm1r_sig]*1e12,'VariableNames',varnames);
disp(Tr);

% ----------------- Bessel filtering ------------------
% Initializing structure fields
npoints=6;
% Here I use an array of structures in order to store
% all the informations relative to each signal
  signal(1).time=time;
  signal(1).I1=I1;
  signal(1).If=I1-I1;
  signal(1).base=[0,0];
  signal(1).fcutoff=0;
  signal(1).Q=0;
  signal(1).tau=[0,0];
  signal(1).Cm=[0,0];
  signal(1).Ra=[0,0];
  signal(1).Rm=[0,0];
  signal(1).fig=figure('Visible',false);

Qstart=tstart; %Transient charge integration limits
Qend=tstart+3*tau;

fcutoff=logspace(0,1,npoints)*1e3; % Set threshold frequencies
fprintf('Filtering %d signals: ', npoints)
for i=1:1:npoints
  % Initializing
  signal(i).time=time;
  signal(i).I1=I1;
  signal(i).If=I1-I1;
  signal(i).fcutoff=fcutoff(i);
  % filtering
  [b, a] = besself(4,fcutoff(i)*2*pi);
  [bz, az] = impinvar(b,a,sr);
  signal(i).If=filter(bz,az,signal(1).I1);
  % fitting
  time_peak=time(signal(i).If == max(signal(i).If));
  tstart_fit=time_peak+0.1e-3;
  [base,base_sig]=baseline(signal(i).If,tstart, tstop, tstep,tau);
  signal(i).base(1)=base;
  signal(i).base(2)=base_sig;
  [tauf,tauf_sig,x,y,fitf]=fitthis(signal(i).time,signal(i).If,tstart_fit, tau, signal(
    i).base(1));
  signal(i).tau=[tauf,tauf_sig];
  % Integrating charge
  Qtransient=signal(i).If(signal(i).time>Qstart & signal(i).time<Qend);
  Qtransient=(Qtransient-signal(i).base(1))*tstep;
  signal(i).Q=sum(Qtransient);
  % Computing parameters
  num=signal(i).tau(1)*Vin_amplitude;
  den=(signal(i).Q + signal(i).tau(1)*signal(i).base(1));
  signal(i).Ra(1)=num/den; % Access resistance
  num_sig=num*tauf_sig/tauf;
  den_sig=sqrt(tauf^2*signal(i).base(2)^2 + signal(i).base(1)^2*tauf_sig^2);
  signal(i).Ra(2)=signal(i).Ra(1)*sqrt((num_sig/num)^2+(den_sig/den)^2);

  signal(i).Rm(1)=Vin_amplitude/signal(i).base(1) - signal(i).Ra(1); %Membrane
    resistance
  signal(i).Rm(2)=sqrt((Vin_amplitude*signal(i).base(2)/signal(i).base(1)^2)^2 + signal
    (i).Ra(2)^2);

  RR=1/signal(i).Ra(1) +1/signal(i).Rm(1); %Membrane capacity
  signal(i).Cm(1)=tauf*RR;
  RR_sig=sqrt((signal(i).Ra(2)/signal(i).Ra(1)^2)^2 + (signal(i).Rm(2)/signal(i).Rm(1)
    ^2)^2);
  signal(i).Cm(2)=signal(i).Cm(1)*sqrt((tauf_sig/tauf)^2 + (RR_sig/RR)^2);
  % Plotting
  signal(i).fig=figure('Visible',false);
  tit='Filtered signal: '+ string(round(fcutoff(i)/1e3,1)) +' kHz';
```

```
  prop = {tit, "Time [ms]", "Current [pA]","m","y"};
  clf
  plot(time*1e3,signal(i).If*1e12);
  hold on
  plot(x*1e3,y*1e12,'b.',x*1e3,fitf*1e12,'r-');
  SetPlot(get(gcf), prop);
  grid on
  filename='Bess'+string(ceil(fcutoff(i)))+'Hz';
  printpdf(signal(i).fig,filename);
  fprintf('%d ',i)

  %results
  Cutfreq(i,1)=signal(i).fcutoff;
  Racc(i,:)=signal(i).Ra;
  Rmem(i,:)=signal(i).Rm;
  Cmem(i,:)=signal(i).Cm;
end
  disp(' ')
  varnames={'Cutoff freq.','Ra [MOhm]','Rm [MOhm]','Cm [pF]'};
  T=table(Cutfreq,Racc/1e6,Rmem/1e6, Cmem*1e12,'VariableNames',varnames);
  disp(T)

% Signal plus filtered signals graph
f70=figure(70);
f70.Visible='off';
f70.Units = 'centimeters';
f70.OuterPosition = [8 8 30 16];
clf
pl1=plot(time*1e3,signal(1).I1*1e12);
pl1.DisplayName='No filtering';
prop = {"Original vs. Filtered Signal", "Time [ms]", "Current [pA]","m","y"};
SetPlot(get(gcf), prop)
legend()
hold on
grid on
for i=1:npoints
  txt= string(round(signal(i).fcutoff/1e3,1)) + ' kHz cutoff';
  plot(time*1e3,signal(i).If*1e12,'DisplayName',txt,'LineWidth',1.2)
end
hold off
printpdf(f70,'Bess_all_freq')

% --------------------- Functions --------------------

function [out_vec] = step_fun(in_vec, time, tstart, tstop, amp)
  % To produce a step function
  in_vec=in_vec-in_vec;
  in_vec(time>=tstart)= amp; %Step function: Vin_amplitude*(theta(tstart)-theta(-tstop)
    )
  in_vec(time>=tstop)=0;
  out_vec=in_vec;
end

function [] = SetPlot(fig, prop )
  %  To set some fancy properties of plots
  font="CMU Serif";
  fontbold= "CMU Serif Bold";
  ax=fig.CurrentAxes;
  ax.Title.String=prop(1);
  ax.Title.FontName=font;
  ax.XLabel.String=prop(2);
  ax.XLabel.FontName=font;
  ax.YLabel.String=prop(3);
  ax.YLabel.FontName=font;

  switch prop{4}
  case "s"
    TitFS=18;
    LabFS=12;
  case "m"
    TitFS=22;
    LabFS=18;
  case "l"
```

```
    TitFS=26;
    LabFS=20;
  otherwise
    TitFS=20;
    LabFS=15;
  end
  ax.Title.FontSize=TitFS;
  ax.XLabel.FontSize=LabFS;
  ax.YLabel.FontSize=LabFS;

  if(size(prop,2)>=5e-3)
    switch prop{5}
    case "y"
      ax.YLim=enlarge(ax.YLim,0.1);
    case "x"
      ax.XLim=enlarge(ax.XLim,0.1);
    case "xy"
      ax.YLim=enlarge(ax.YLim,0.1);
      ax.XLim=enlarge(ax.XLim,0.1);
    otherwise
      return
    end
  end
end

function [outint]=enlarge(inint,hwmuch)
  % To set fancy Y limits to plots
  interv= abs(inint(2)-inint(1));
  piece= 0.5*(hwmuch*interv);
  if inint(1)<1e-12
    outint= [inint(1), inint(2)+piece];
  else
    outint= [inint(1)-piece, inint(2)+piece];
  end
end

function [outint]=pushup(inint,hwmuch)
  % To higher data on a plot
  interv= abs(inint(2)-inint(1));
  piece= 0.5*(hwmuch*interv);
  outint= [inint(1)-piece, inint(2)-piece];
end

function [peak_val]=PeakEstim(time,signal,peak_time,mean_int)
  % This function performs a mean of the values of
  % the vector signal from time_loc to time_loc+mean_int
  timest=time(2)-time(1);
  peak_idx=ceil(peak_time/timest)+1;
  peak_val=mean(signal(peak_idx:peak_idx+mean_int));
end

function []=printpdf(fig_handle,filename)
  % To save figures in pdf
  set(fig_handle,'Units','Inches');
  pos = get(fig_handle,'Position');
  set(fig_handle,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize',[pos(3),
    pos(4)])
  print(fig_handle,filename,'-dpdf','-bestfit');

end

function [I1_infty, I1_infty_sig]=baseline(I1, tstart, tstop, tstep, tau)
  % To compute baselines of signals
  I1_infty_start = ceil((tstart+5*tau)/tstep);  %I1_infty estimation from data
  I1_infty_end = ceil(tstop/tstep)-10;
  I1_infty = mean(I1(I1_infty_start:I1_infty_end));
  how_many=sqrt(I1_infty_end-I1_infty_start);
  I1_infty_sig=std(I1(I1_infty_start:I1_infty_end))/how_many;
end


function [tau1,tau1_sig,x, y, fitfun]=fitthis(xdata, ydata, tstart_fit, tau, baseline)
  % To fit exponential data
```

```
  x= xdata(xdata>tstart_fit & xdata<tstart_fit+4*tau); % Select right time interval
  y= ydata(xdata>tstart_fit & xdata<tstart_fit+4*tau) - baseline; % subtract baseline
  fitres=fit(x,y,'exp1','startpoint',[1e-9,-1/0.2545e-3]); % fit specifying initial
    parameters
  tau1=-1/fitres.b;
  bounds=confint(fitres);
  bounds=-(1./bounds);
  bounds(:,1)=[];
  tau1_sig=(bounds(2)-bounds(1))/2;
  y=y+baseline;
  fitfun=fitres(x)+baseline;
end
```