Modelling Action Potentials

Introduction

In this section we will use a model of voltage-gated ion channels in a single neuron to simulate action potentials. The model is based on the work by Hodgkin & Huxley in the 1940s and 1950s [1][2]. A good reference to refresh your memory about how ion channels in a neuron work is the Kandel, Schwartz & Jessel book "Principles of Neural Science" [4].

To model the action potential we will use an article by Ekeberg et al. (1991) published in Biological Cybernetics [3]. When reading the article you can focus on the first three pages (up to paragraph 2.3) and try to find answers to the following questions:

- How many differential equations are there?
- What is the order of the system described in equations 1-9?
- What are the states and state derivatives of the system?

Simulating the Hodgkin & Huxley model

Before we begin coding up the model, it may be useful to remind you of a fundamental law of electricity, one that relates electrical potential V to electric current I and resistance R (or conductance G, the reciprocal of resistance). This of course is known as Ohm's law:

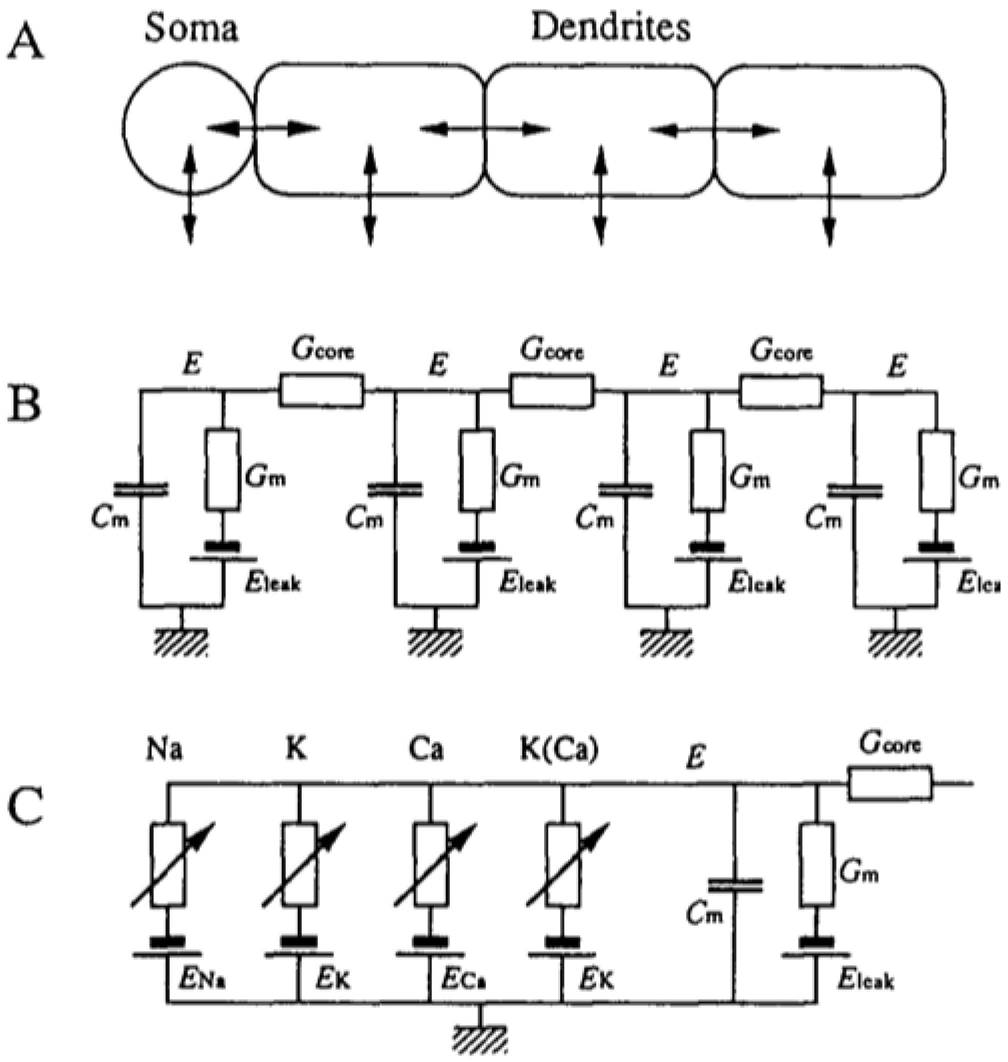
$V=IR$

or

$V=IG$

Our goal here is to code up a dynamical model of the membrane's electric circuit including two types of ion channels: sodium and potassium channels. We will use this model to better understand the process underlying the origin of an action potential.

The neuron model



Schematic of Ekeberg et al. 1991 neuron model

The figure above, adapted from Ekeberg et al., 1991 [3], schematically illustrates the model of a neuron. In panel A we see a soma, and multiple dendrites. Each of these can be modelled by an electrical "compartment" (Panel B) and the passive interactions between them can be modelled as a pretty standard electrical circuit (see Biological Neuron Model for more details about compartmental models of neurons). In panel C, we see an expanded model of the Soma from panel A. Here, a number of active ion channels are included in the model of the soma.

For our purposes here, we will focus on the soma, and we will not include any additional dendrites in our implementation of the model. Thus essentially we will be modelling what appears in panel C, and at that, only a subset.

In panel C we see that the soma can be modelled as an electrical circuit with a sodium ion channel (Na), a potassium ion channel (K), a calcium ion channel (Ca), and a calcium-dependent potassium channel (K(Ca)). What we will be concerned with simulating, ultimately, is the intracellular potential E.

Passive Properties

Equation (1) of Ekeberg is a differential equation describing the relation between the time derivative of the membrane potential E as a function of the passive leak current through the membrane, and the current through the ion channels. Note that Ekeberg uses E instead of the typical V symbol to represent electrical potential.

$dEdt=(Eleak-E)Gm+\Sigma(Ecomp-E)Gcore+IchannelsCm$

Don't panic, it's not actually that complicated. What this equation is saying is that the rate of change of electrical potential across the current (the left hand side of the equation, $dEdt$) is equal to the sum of a bunch of other terms, divided by membrane capacitance Cm (the right hand side of the equation). Recall from basic physics thatcapacitance is a measure of the ability of something to store an electrical charge.

The "bunch of other things" is a sum of three things, actually, (from left to right): a passive leakage current, plus a term characterizing the electrical coupling of different compartments, plus the currents of the various ion channels. Since we are not going to be modelling dendrites here, we can ignore the middle term on the right hand side of the equation $\Sigma(Ecomp-E)Gcore$ which represents the sum of currents from adjacent compartments (we have none).

We are also going to include in our model an external current $Iext$. This can essentially represent the sum of currents coming in from the dendrites (which we are not explicitly modelling). It can also represent external current injected in a patch-clamp experiment. This is what we as experimenters can manipulate, for example, to see how neuron spiking behaviour changes. So what we will actually be working with is this:

$dEdt=(Eleak-E)Gm+Ichannels+IextCm$

What we need to do now is unpack the $Ichannels$ term representing the currents from all of the ion channels in the model. Initially we will only be including two, the potassium channel (K) and the sodium channel (Na).

Sodium channels (Na)

The current through sodium channels that enter the soma are represented by equation (2) in Ekeberg et al. (1991):

$INa=(ENa-Esoma)GNam3h$

where m is the activation of the sodium channel and h is the inactivation of the sodium channel, and the other terms are constant parameters: ENa is the reversal potential, GNa is the maximum sodium conductance throught the membrane, and $Esoma$ is the membrane potential of the soma.

The activation m of the sodium channels is described by the differential equation (3) in Ekeberg et al. (1991):

$dmdt=\alpha m(1-m)-\beta mm$

where αm represents the rate at which the channel switches from a closed to an open state, and βm is rate for the reverse. These two parameters α and β depend on the membrane potential in the soma. In other words the sodium channel is voltage-gated. Equation (4) in Ekeberg et al. (1991) gives these relationships:

$\alpha m\beta m==A(Esoma-B)1-e(B-Esoma)/CA(B-Esoma)1-e(Esoma-B)/C$

A tricky bit in the Ekeberg et al. (1991) paper is that the A, B and C parameters above are different for α and β even though there is no difference in the symbols used in the equations.

The inactivation of the sodium channels is described by a similar set of equations: a differential equation giving the rate of change of the sodium channel deactivation, from Ekeberg et al. (1991) equation (5):

$dhdt=\alpha h(1-h)-\beta hh$

and equations specifying how αh and βh are voltage-dependent, given in Ekeberg et al. (1991) equation (6):

$\alpha h\beta h==A(B-Esoma)1-e(Esoma-B)/CA1-e(B-Esoma)/C$

Note again that although the terms *A*, *B* and *C* are different for αh and βh even though they are represented by the same symbols in the equations.

So in summary, for the sodium channels, we have two state variables: (m,h) representing the activation (m) and deactivation (h) of the sodium channels. We have a differential equation for each, describing how the rate of change (the first derivative) of these states can be calculated: Ekeberg equations (3) and (5). Those differential equations involve parameters (α,β), one set for m and a second set for h. Those (α,β) parameters are computed from Ekeberg equations (4) (for m) and (6) (for h). Those equations involve parameters (A,B,C) that have parameter values specific to α and β and m and h (see Table 1 of Ekeberg et al., 1991).

Potassium channels (K)

The potassium channels are repreested in a similar way, although in this case there is only channel activation, and no inactivation. In Ekeberg et al. (1991) the three equations (7), (8) and (9) represent the potassium channels:

$$Ik=(Ek-Esoma)Gkn4$$

$$dndt=\alpha n(1-n)-\beta nn$$

where n is the state variable representing the activation of potassium channels. As before we have expressions for (α,β) which represent the fact that the potassium channel is also voltage-gated:

$$\alpha n\beta n==A(Esoma-B)1-e(B-Esoma)/CA(B-Esoma)1-e(Esoma-B)/C$$

Again, the parameter values for (A,B,C) can be found in Ekeberg et al., (1991) Table 1.

To summarize, the potassium channel has a single state variable n representing the activation of the potassium channel.

Summary

We have a model now that includes four state variables:

1. E representing the potential in the soma, given by differential equation (1) in Ekeberg et al., (1991)
2. m representing the activation of sodium channels, Ekeberg equation (3)

3. *h* representing the inactivation of sodium channels, Ekeberg equation (5)
4. *n* representing the activation of potassium channels, Ekeberg equation (8)

Each of the differential equations that define how to compute state derivatives, involve (α,β) terms that are given by Ekeberg equations (4) (for *m*), (6) (for *h*) and (9) (for *n*).

So what we have to do in order to simulate the dynamic behaviour of this neuron over time, is simply to implement these equations in Python code, give the system some reasonable initial conditions, and simulate it over time using the odeint() function.

Python code

A full code listing of a model including sodium and potassium channels can be found here: ekeberg1.py. Admittedly, this system involves more equations, and more parameters, than the other simple "toy" systems that we saw in the previous section. The fundamental ideas are the same however, so let's step through things bit by bit.

We begin by setting up all of the necessary model parameters (there are many). They are found in Ekeberg et al. (1991) Tables 1 and 2. I have chosen to do this using a Python data type called a dictionary. This is a useful data type to parcel all of our parameters together. Unlike an array or list, which we would have to index using integer values (and then keep track of which one corresponded to which parameter), with a dictionary, we can index into it using string labels.

```
# ipython --pylab

# import some needed functions
from scipy.integrate import odeint

# set up a dictionary of parameters

E_params = {
	'E_leak' : -7.0e-2,
	'G_leak' : 3.0e-09,
	'C_m'    : 3.0e-11,
	'I_ext'  : 0*1.0e-10
}
```

```
Na_params = {
        'Na_E'          : 5.0e-2,
        'Na_G'          : 1.0e-6,
        'k_Na_act'      : 3.0e+0,
        'A_alpha_m_act' : 2.0e+5,
        'B_alpha_m_act' : -4.0e-2,
        'C_alpha_m_act' : 1.0e-3,
        'A_beta_m_act'  : 6.0e+4,
        'B_beta_m_act'  : -4.9e-2,
        'C_beta_m_act'  : 2.0e-2,
        'l_Na_inact'    : 1.0e+0,
        'A_alpha_m_inact' : 8.0e+4,
        'B_alpha_m_inact' : -4.0e-2,
        'C_alpha_m_inact' : 1.0e-3,
        'A_beta_m_inact'  : 4.0e+2,
        'B_beta_m_inact'  : -3.6e-2,
        'C_beta_m_inact'  : 2.0e-3
}

K_params = {
        'k_E'           : -9.0e-2,
        'k_G'           : 2.0e-7,
        'k_K'           : 4.0e+0,
        'A_alpha_m_act' : 2.0e+4,
        'B_alpha_m_act' : -3.1e-2,
        'C_alpha_m_act' : 8.0e-4,
        'A_beta_m_act'  : 5.0e+3,
        'B_beta_m_act'  : -2.8e-2,
        'C_beta_m_act'  : 4.0e-4
}

params = {
        'E_params'  : E_params,
        'Na_params' : Na_params,
        'K_params'  : K_params
}
```

We could have stored the four values in E_params in an array like this:

```
E_params = array([-7.0e-2, 3.0e-09, 6.0e-11, 0*1.0e-10])
```

but then we would have to access particular values by indexing into that array with integers like this:

```
E_leak = E_params[0]
G_leak = E_params[1]
```

Instead, if we use a dictionary, we can index into the structure using alphanumeric strings as index values, like this:

```
E_params['E_leak']
E_params['G_leak']
```

You don't have to use a dictionary to store the parameter values, but I find it a really useful way to maintain readability.

Our next bit of code is the implementation of the ODE function itself. Remember, our ultimate goal is to model *E*, the potential across the soma membrane. We know from Ekeberg equation (1) that the rate of change of *E* depends on leakage current and on the sum of currents from other channels. These other currents are given by Ekeberg equations (2) (sodium) and (7) (potassium). These equations involve the three other states in our system: sodium activation *m*, sodium inactivation *h* and potassium activation *n*, which are each defined by their own differential equations, Ekeberg equations (3), (5) and (8), respectively. It's just a matter of coding things up step by step.

```
# define our ODE function

def neuron(state, t, params):
        """
         Purpose: simulate Hodgkin and Huxley model for the action potential using
         the equations from Ekeberg et al, Biol Cyb, 1991.
         Input: state ([E m h n] (ie [membrane potential; activation of
                   Na++ channel; inactivation of Na++ channel; activation of K+
                   channel]),
                 t (time),
                 and the params (parameters of neuron; see Ekeberg et al).
         Output: statep (state derivatives).
        """

        E = state[0]
        m = state[1]
        h = state[2]
        n = state[3]

        Epar = params['E_params']
        Na   = params['Na_params']
        K    = params['K_params']

        # external current (from "voltage clamp", other compartments, other neurons, etc)
        I_ext = Epar['I_ext']

        # calculate Na rate functions and I_Na
        alpha_act = Na['A_alpha_m_act'] * (E-Na['B_alpha_m_act']) / (1.0 - 
exp((Na['B_alpha_m_act']-E) / Na['C_alpha_m_act']))
        beta_act = Na['A_beta_m_act'] * (Na['B_beta_m_act']-E) / (1.0 - exp((E-
Na['B_beta_m_act']) / Na['C_beta_m_act']) )
        dmdt = ( alpha_act * (1.0 - m) ) - ( beta_act * m )

        alpha_inact = Na['A_alpha_m_inact'] * (Na['B_alpha_m_inact']-E) / (1.0 - exp((E-
Na['B_alpha_m_inact']) / Na['C_alpha_m_inact']))
        beta_inact  = Na['A_beta_m_inact'] / (1.0 + (exp((Na['B_beta_m_inact']-E) / 
Na['C_beta_m_inact'])))
        dhdt = ( alpha_inact*(1.0 - h) ) - ( beta_inact*h )

        # Na-current:
        I_Na =(Na['Na_E']-E) * Na['Na_G'] * (m**Na['k_Na_act']) * h

        # calculate K rate functions and I_K
        alpha_kal = K['A_alpha_m_act'] * (E-K['B_alpha_m_act']) / (1.0 - 
exp((K['B_alpha_m_act']-E) / K['C_alpha_m_act']))
```

```
        beta_kal = K['A_beta_m_act'] * (K['B_beta_m_act']-E) / (1.0 - exp((E-K['B_beta_m_act']) / K['C_beta_m_act']))
        dndt = ( alpha_kal*(1.0 - n) ) - ( beta_kal*n )
        I_K = (K['k_E']-E) * K['k_G'] * n**K['k_K']

        # leak current
        I_leak = (Epar['E_leak']-E) * Epar['G_leak']

        # calculate derivative of E
        dEdt = (I_leak + I_K + I_Na + I_ext) / Epar['C_m']
        statep = [dEdt, dmdt, dhdt, dndt]

        return statep
```

Next we run a simulation by setting up our initial states, and a time array, and then calling odeint(). Note that we are injecting some external current by changing the value of the params['E_params']['I_ext'] entry in the params dictionary.

```
# simulate

# set initial states and time vector
state0 = [-70e-03, 0, 1, 0]
t = arange(0, 0.2, 0.001)

# let's inject some external current
params['E_params']['I_ext'] = 1.0e-10

# run simulation
state = odeint(neuron, state0, t, args=(params,))
```

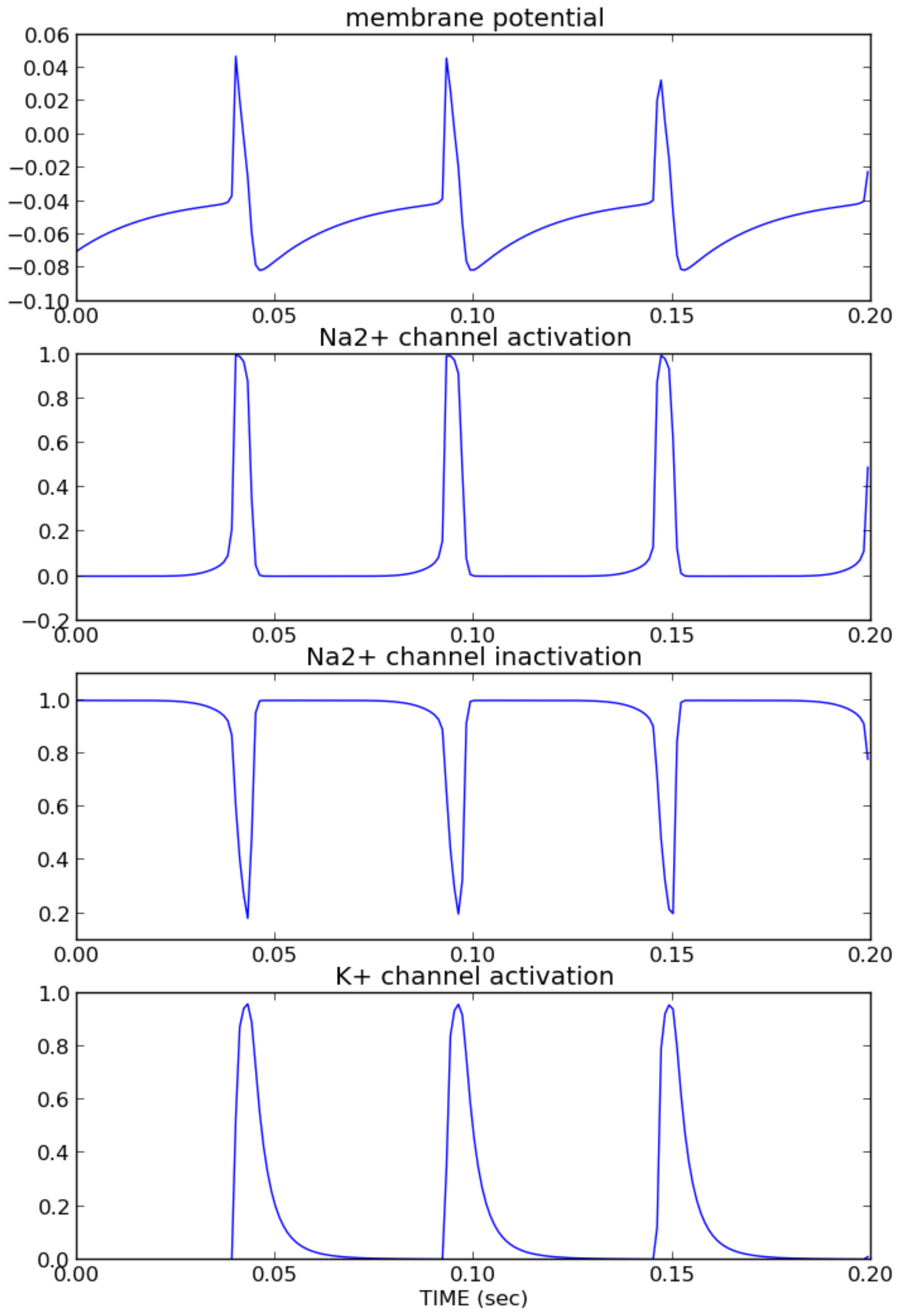
Finally, we plot the results:

```
# plot the results

figure(figsize=(8,12))
subplot(4,1,1)
plot(t, state[:,0])
```

```
title('membrane potential')
subplot(4,1,2)
plot(t, state[:,1])
title('Na2+ channel activation')
subplot(4,1,3)
plot(t, state[:,2])
title('Na2+ channel inactivation')
subplot(4,1,4)
plot(t, state[:,3])
title('K+ channel activation')
xlabel('TIME (sec)')
```

Here is what you should see:



Spiking neuron simulation based on Ekeberg et al., 1991

Things to try

1. alter the ekeberg1.py code so that the modelled neuron only has the leakage current and external current. In other words, comment out the terms related to sodium and potassium channels. Run a simulation with an initial membrane potential of -70mv and an external current of 0.0mv. What happens and why?
2. Change the external current to 1.0e-10 and re-run the simulation. What happens and why?
3. Add in the terms related to the sodium channel (activation and deactivation). Run a simulation with external current of 1.0e-10 and initial states [-70e-03, 0, 1]. What happens and why?
4. Add in the terms related to the potassium channel. Run a simulation with external current of 1.0e-10 and initial states [-70e-03, 0, 1, 0]. What happens and why?
5. Play with the external current level (increase it slightly, decrease it slightly, etc). What is the effect on the behaviour of the neuron?
6. What is the minimum amount of external current necessary to generate an action potential? Why?

Next steps

Next we will be looking at models of motor control. We will be using human arm movement as the model system. We will first look at kinematic models of one and two-joint arms, so we can talk about the problem of coordinate transformations between hand-space and joint-space, and the non-linear geometrical transformations that must take place. After that we will move on to talking about models of muscle, force production, and limb dynamics, with an eye towards a modelling the neural control of arm movements such as reaching and pointing.

[Computational Motor Control: Kinematics]

References

[1] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (Lond.)*, 117(4):500-544, Aug 1952. [bib]

[2] A. L. Hodgkin, A. F. Huxley, A. L. Hodgkin, and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. 1952.*Bull. Math. Biol.*, 52(1-2):25-71, 1990. [bib]

[3] O. Ekeberg, P. Wallen, A. Lansner, H. Traven, L. Brodin, and S. Grillner. A computer based model for realistic simulations of neural networks. I. The single neuron and synaptic interaction. *Biol Cybern*, 65(2):81–90, 1991. [bib]

[4] E.R. Kandel, J.H. Schwartz, T.M. Jessell, et al. *Principles of neural science*, volume 4. McGraw-Hill New York, 2000. [bib]

Source:

http://www.gribblelab.org/compneuro2012/3_Modelling_Action_Potentials.html