Tutorial 2: More Examples

Example 6. Voltage Clamp

We are going to build a single-compartment neuron to see how the voltage clamp works. First, we prepare our cell:

```
>>> from PyHH import *
>>> import pylab as plt
>>> soma = Compartment(diameter = 50, length=None)
>>> soma.add_channels([NaC, KDR, gL])
```

Now we are going to define a voltage clamper from VClamper:

```
>>> clp = VClamper()
>>> clp.Waveform = Rect(delay=1, width=5, amplitude=40)
>>> clp.connect(soma)
```

The default baseline is -60 mV. If you don't like this baseline value, for example, you can set a new one by clp.set_baseline(-70). Now we just use the default value. It's time to define and run an experiment:

```
>>> xp = Experiment(soma)
>>> xp.run(10, 0.005)
```

It takes less time to get results in voltage-clamp simulation than in the current clamp simulation. We know that voltage clamp experiments record currents. In PyHH, the current is called Jp and stored in the clamper.

```
>>> plt.figure()
>>> plt.subplot(3,1,1)
>>> plt.plot(xp.T, clp.Jp)
>>> plt.ylabel('current (pA/um2)')
>>> plt.ylim([-5,5])
```

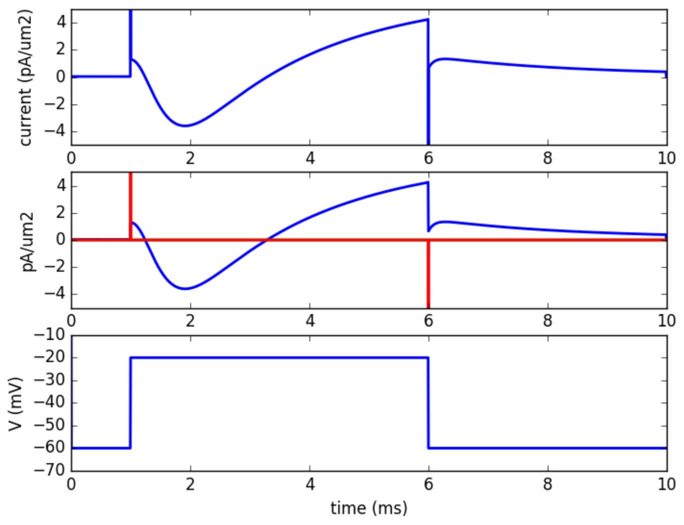
During real patch clamp experiments, we want to measure the transmembrane current (Jm), and we always suppose that Jp = Jm. Actually even for ideal voltage clampers and ideal pipets, Jp is not equal to Jm. Jp is composed of at least two components, the transmembrane capacity current (Jc) and the transmembrane ionic current (Jm). If space clamp is not good, the cytosolic current (Jn) also contributes to Jp. If Jn is small, we can expect Jp ≈ Jm. In real experiment, we normally see Jp, not the components of Jp. The PyHH VClamper stores three currents in the form of current density. So we are going to show Jm and Jc as well.

```
>>> plt.subplot(3,1,2)
>>> plt.plot(xp.T, clp.Jm, linewidth=2.0) # this is what people want in real experiments
>>> plt.plot(xp.T, clp.Jn, linewidth=2.0) # should be zero
>>> plt.plot(xp.T, clp.Jc, linewidth=2.0) # the upward and downward spikes
>>> plt.ylabel('pA/um2')
>>> plt.ylim([-5,5])
```

And finally we show the membrane potential of the compartment, which is just the command potential:

```
>>> plt.subplot(3,1,3)
>>> plt.plot(xp.T, soma.Vm, linewidth=2.0)
>>> plt.xlabel('time (ms)')
>>> plt.ylabel('V (mV)')
>>> plt.ylim([-65,-25])
>>> plt.show()
```

You will see something like:



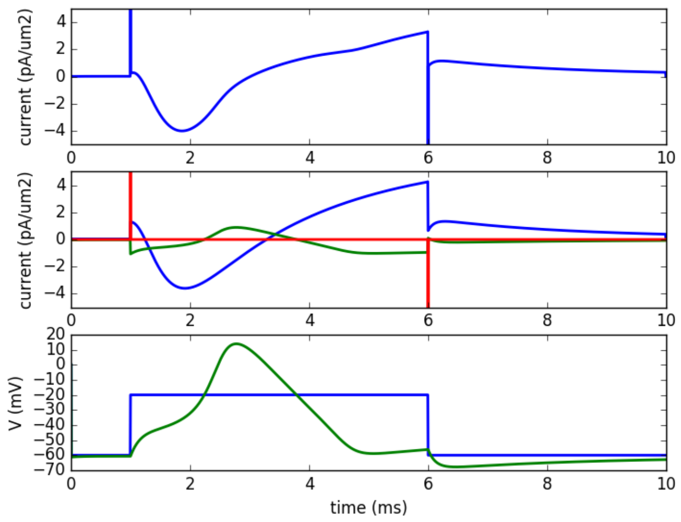
The top panel shows Jp, which is the sum of Jc and Jm. Jn is zero in this example. The middle panel show Jm (in blue) and Jc (in red). Recording from a multi-compartment neuron will be more complicated due to space clamp problem.

Example 7. Space Clamp

In this example, we are going to learn something about space clamp, which has been ignored by many patch clampers. The example codes are very similar to the previous one, but we use two compartments.

```
>>> from PyHH import *
>>> import pylab as plt

>>> soma = Compartment(diameter = 50, length=None)
>>> soma.add_channels([NaC, KDR, gL])
>>> dend = Compartment(diameter = 1.5, length = 100)
>>> dend.add_channels([NaC, KDR, gL])
>>> soma.connect(dend)

>>> clp = VClamper()
>>> clp.Waveform = Rect(delay=1, width=5, amplitude=40)
>>> clp.connect(soma)
>>> xp = Experiment([soma,dend])
>>> xp.run(10, 0.005)

>>> plt.figure()
>>> plt.subplot(3,1,1)
>>> plt.plot(xp.T, clp.Jp, linewidth=2.0)
>>> plt.ylabel('current (pA/um2)')
>>> plt.ylim([-5,5])
>>> plt.subplot(3,1,2)
>>> plt.plot(xp.T, clp.Jm, linewidth=2.0)
>>> plt.plot(xp.T, clp.Jn, linewidth=2.0)
>>> plt.plot(xp.T, clp.Jc, linewidth=2.0)
>>> plt.ylabel('current (pA/um2)')
>>> plt.ylim([-5,5])

>>> plt.subplot(3,1,3)
>>> plt.plot(xp.T, soma.Vm, linewidth=2.0)
>>> plt.plot(xp.T, dend.Vm, linewidth=2.0)
>>> plt.xlabel('time (ms)')
>>> plt.ylabel('V (mV)')
>>> plt.show()
```

Now you see things are quite different. In the following figure, the blue curve in the top panel is the current picked up by the pipet, namely, Jp. However, the blue curve in the middle channel is what we want (Jm). We can see the difference between Jp and Jm, which is caused by the cytosolic current (green curve in the middle panel) flowing between different parts of the neuron, namely between the compartments soma and dend.



What happens if we replace the line:

clp.connect(soma)

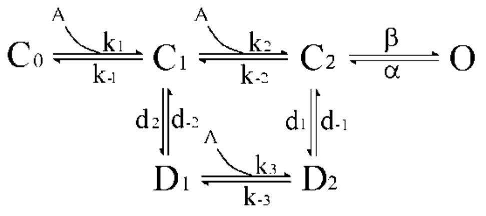
with

clp.connect([soma,dend])

I am pretty sure that you know the answer.

Example 8: the Ligand-Gated Ion Channel

In this example, I will use AMPA receptors to illustrate the simulation of LGICs based on Markov transition schemes. There are many transition schemes available, currently I use the scheme from Krampfl et al., 2002 in Eur. J. Neurosci. 15:51-62.

	In this scheme, C represents closed states, O represents open states, and D represents deactivation states. k, α, β and d are rate constants.

Before defining the AMPA receptor, we have to define glutamate, the ligand of the AMPAR:

```
>>> Glu = Ligand()
```

I use Python dictionaries to represent the transition graph:

```
>>> ampar_transit = {
                        'C0': {'C1': 4.0},
                        'C1': {'C0': 2.0,   'D1': 0.15, 'C2': 2.0},
                        'C2': {'C1': 4.0,   'D2': 0.70, 'O': 20.0},
                        'D1': {'C1': 0.015, 'D2': 2.0},
                        'D2': {'C2': 0.002, 'D1': 0.875},
                        'O':  {'C2': 8.0}
                       }
```

You will understand this dictionary if you compare it with the transition scheme. In addition, we have to tell where the agonist (ligand) bind, and I also use Python dictionaries for this purpose:

```
>>> ampar_binding = {Glu:
                         {'C0': 'C1',
                          'C1': 'C2',
                          'D1': 'D2'
                         }
                        }
```

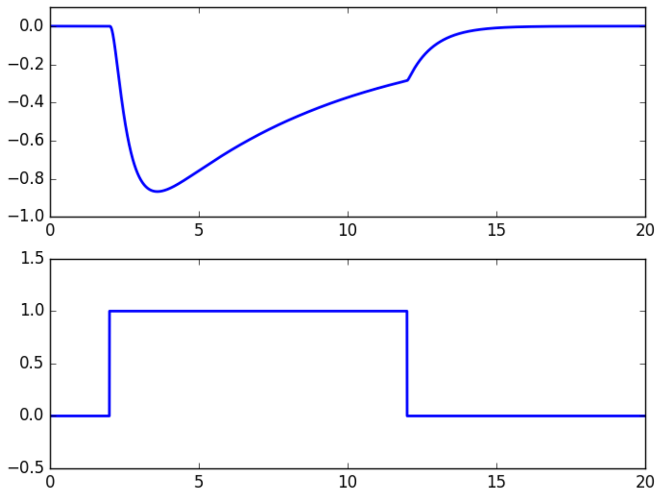
Now we can define the AMPA receptor:

```
>>> AMPAR = LGIC(ampar_transit, ampar_binding, gMax = 0.05, ER = 0)
```

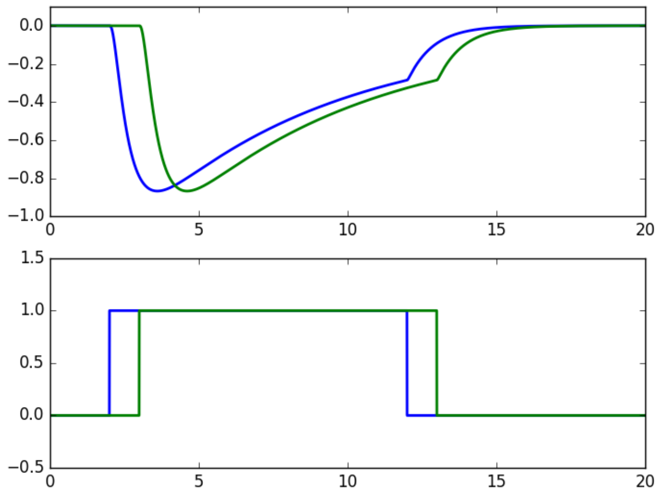
The complete codes are as follows:

```
>>> from PyHH import *
>>> import pylab as plt

>>> Glu = Ligand()
>>> ampar_transit = {
                                   'C0': {'C1': 4.0},
                                   'C1': {'C0': 2.0,   'D1': 0.15, 'C2': 2.0},
                                   'C2': {'C1': 4.0,   'D2': 0.70, 'O': 20.0},
                                   'D1': {'C1': 0.015, 'D2': 2.0},
                                   'D2': {'C2': 0.002, 'D1': 0.875},
                                   'O':  {'C2': 8.0}
                                  }

>>> ampar_binding = {Glu:
                                     {'C0': 'C1',
                                      'C1': 'C2',
                                      'D1': 'D2'
                                      }
                                   }
>>> AMPAR = LGIC(ampar_transit, ampar_binding, gMax = 0.05, ER = 0)
>>> cpm = Compartment(diameter = 1.5, length = 100)
>>> ampar = cpm.add_channel(AMPAR)
>>> cpm.add_channel(gL)
>>> vclp = VClamper()
>>> vclp.Waveform = Rect(delay=0, width=150, amplitude=0)
>>> vclp.connect(cpm)
>>> deliver = CClamper(ampar.Ligand)
>>> deliver.Waveform = Rect(delay=2, width=10, amplitude=1)
>>> deliver.connect(cpm)
>>> xp = Experiment(cpm)
>>> xp.run(20,dt=0.005)

>>> plt.figure()
>>> plt.subplot(2,1,1)
>>> plt.plot(xp.T, vclp.Jm, linewidth=2.0)
>>> plt.ylim([-1,0.1])
>>> plt.subplot(2,1,2)
>>> plt.plot(xp.T, deliver.Command, linewidth=2.0)
>>> plt.ylim([-0.5,1.5])
>>> plt.show()
```

You will see something like:



The file example9.py provides another example for AMPAR modeling with two compartments. Run the script and you will see something like:



In next tutorial, I will go back to the voltage-gated ion channels and explain how NaC, KDR and gL are defined.