

## Tutorial 1: the Basic

PyHH is a small library of Python for simulation of the electrical activities of neurons. PyHH contains a few classes, which correspond to experiment objects in real experiments. The most basic classes are Compartment, NaChannel, KChannel, LeakChannel, Ligand, LGIC (ligand-gated ion channel), IClamper (for current injection), VClamper (for voltage clamp), CClamper (for concentration clamp) and Experiment (for integration of the equations). Doing a simulation with these classes is just like performing a real experiment in a lab. Since PyHH is not standalone software, you need to write a Python script and run it from a terminal, which is called the command line in Windows system. Python is pre-installed on Linux or Mac systems. Windows, however, requires the user to install Python. Just go to the official Python website and download the latest python (Python 2.7.13 or Python 3.6.1) and install it with a few clicks. Python 2.7.x will be installed in the path C:\Python27. So you can launch a command line and start Python by typing:

```
C:\Python27\python
```

You can run a Python script (say example2.py) by typing:

```
C:\Python27\python example2.py
```

If you let Windows know where Python is by setting the environmental variables, you can just type

```
python
```

in order to start Python

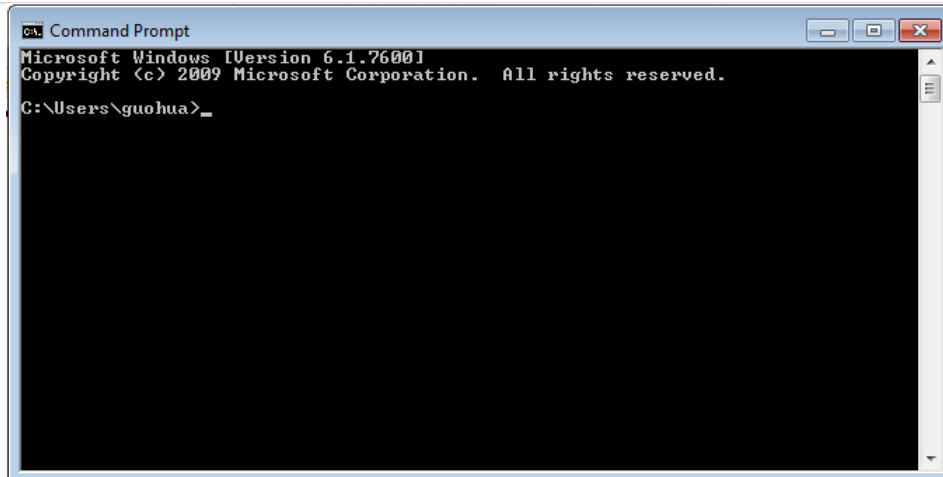
or

```
python example2.py
```

to run the Python script example2.py

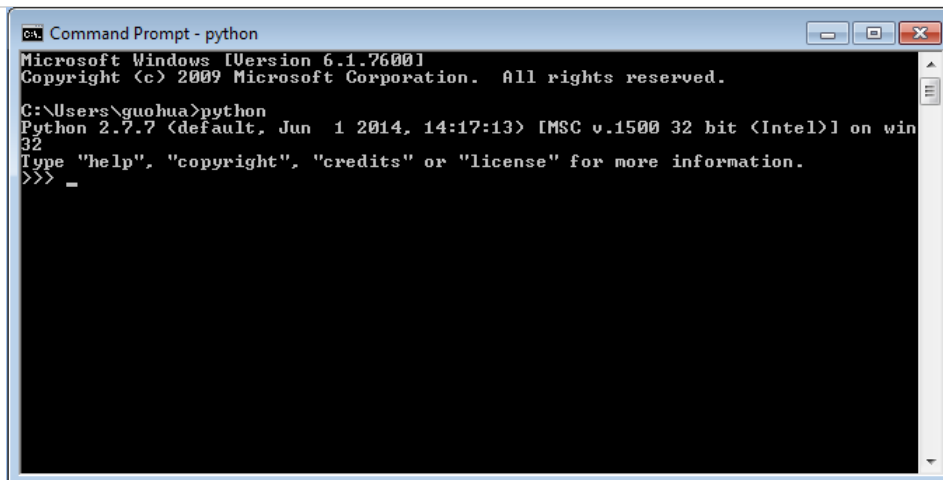
Check on the internet to see how to set the environmental variables. I had some bad experience.

I don't use Windows, but to test PyHH on Windows, I had installed Python and Matplotlib on my dual boot computer. See the link <https://docs.python.org/2/faq/windows.html> for how to run Python programs on Windows. On my Windows system, the command Prompt looks like



```
cmd - Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\guohua>_
```

After I type python, I get Python interpreter in the interactive mode:



```
cmd - Command Prompt - python
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\guohua>python
Python 2.7.7 (default, Jun 1 2014, 14:17:13) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

I suggest trying PyHH in the interactive mode first.

You don't need to install PyHH, just drop pyhh.py in a folder where you want to use it. But, you need something, such as Matplotlib, for visualization of the simulation results. For the installation of Matplotlib, you can download a python file named get-pip.py and run the script by typing:

```
python get-pip.py
```

then run

```
python -m pip install -U pip setuptools
```

```
python -m pip install matplotlib
```

Done. There are other ways to install matplotlib.

From now on, I suppose that you already start a command prompt.

I keep pyhh.py in a folder called PyHH, to move into this directory, type:

```
cd PyHH
```

Now let's start Python by typing:

```
python
```

or

```
python3
```

### **Example 1. A Single Compartment Model**

We are going to start with PyHH basic classes. First, import everything from PyHH.

```
>>> from pyhh import *
```

Now you can define a compartment from the Compartment class:

```
>>> cpm = Compartment(diameter = 1.5, length = 100)
```

or simply: `cpm = Compartment(1.5, 100)`

Now you define a cylinder with defined diameter and length. The unit is **micrometer**. Now you can add channels to the compartment:

```
>>> cpm.add_channels([NaC, KDR, gL])
```

NaC, KDR and gL are predefined conductances for the sodium channel, potassium channel and leak channel, respectively. You will learn how to define the conductances later. Now feed the compartment to an experiment:

```
>>> xp = Experiment(cpm)
```

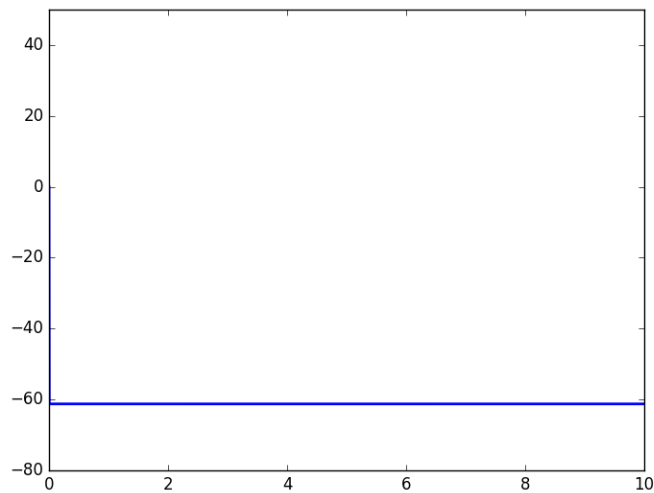
This defines an experiment, which gathers information from the compartment for integration of equations.

```
>>> xp.run(t=10, dt=0.005)
```

This will make a 10-ms simulation run with time step = 0.005 ms. The solution for membrane potential is stored in `cpm.Vm`, and the time series in `xp.T`. You can use your favourite software tool to plot the T-Vm curve. Currently I use `matplotlib`:

```
>>> import pylab as plt    # import the tool
>>> plt.figure()           # create a figure
>>> plt.subplot(1,1,1)     # add a plot
>>> plt.plot(xp.T, cpm.Vm) # plot it
>>> plt.ylim([-80,50])     # set y-range, namely, Vm range
>>> plt.show()             # show the plot
```

You will see something like



The result is disappointing, but we missed something. Now, let's have a look at Example 2.

## Example 2. Current Clamp

In this example, we define a clamper for current injection.

```
>>> from pyhh import *
>>> import pylab as plt
>>> cpm = Compartment(diameter = 1.5, length = 100)
>>> cpm.add_channels([NaC, KDR, gL])
```

Define a current clamper:

```
>>> clp = IClamper()
```

Set the current waveform, a rectangular one:

```
>>> clp.Waveform = Rect(delay=1, width=1, amplitude=0.7)
```

Connect the clamper to the compartment:

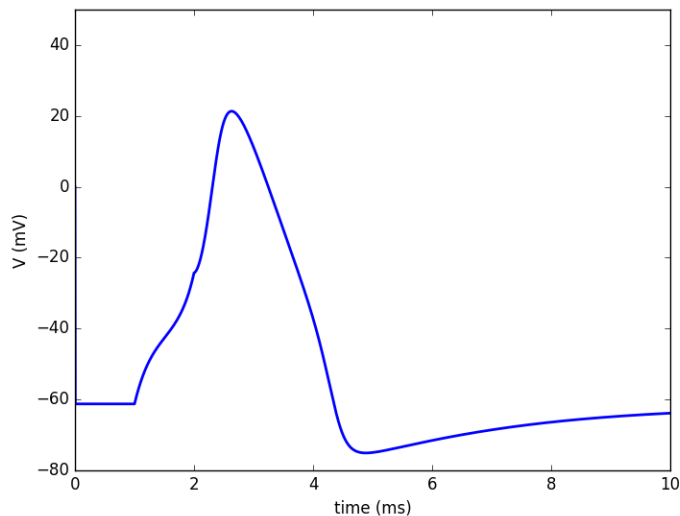
```
>>> clp.connect(cpm)
```

Do the experiment as before:

```
>>> xp = Experiment([cpm])
>>> xp.run(10,dt=0.005)
```

```
>>> plt.figure()
>>> plt.subplot(1,1,1)
>>> plt.plot(xp.T, cpm.Vm, linewidth=2.0)
>>> plt.ylim([-80,50])
>>> plt.show()
```

Now you see something interesting.



Please note the current amplitude is in the unit of  $\text{pA}/\mu\text{m}^2$ . This means that the amplitude is current density. The absolute current values can be easily calculated by

Current = amplitude \* cpm.Surface

### Example 3. Spherical Compartment

A sphere has no length. So you can define a sphere by

```
>>> cpm = Compartment(diameter = 50, length = None)
```

Here length=None. PyHH will treat the compartment as a sphere. Try ex3.py by yourself.

### Example 4. Two Compartments

From this example, we are going to deal with neurons with multiple compartments.

```
>>> from pyhh import *
>>> import pylab as plt
>>> soma = Compartment(50,length=None) # please note I did not define length for soma.
>>> dend = Compartment(1.5,length=100)
>>> soma.add_channels([NaC, KDR, gL])
>>> dend.add_channels([NaC, KDR, gL])
>>> soma.connect(dend)
```

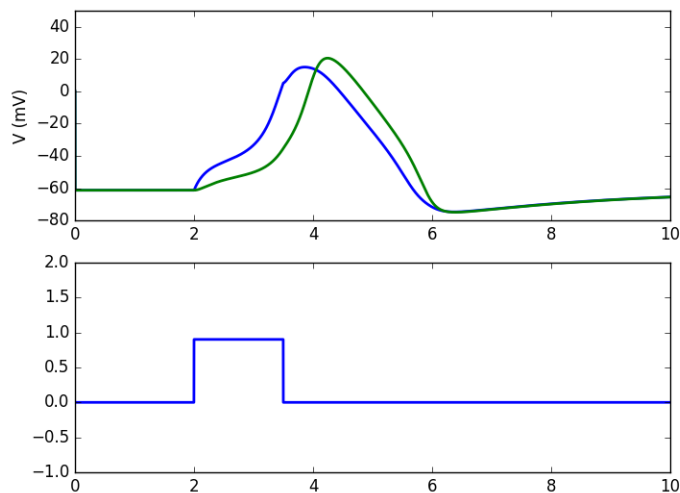
Here is the trick, you can not do dend.connect(soma). Connection is directed in PyHH. I define a function which perform the same task: dend.attached\_to(soma)

```
>>> clp = IClamper()
>>> clp.Waveform = Rect(delay=2, width=1.5, amplitude=1.)
>>> clp.connect(soma)
>>> xp = Experiment([soma, dend]) # the compartments are supplied as a list
```

If you have more than one compartment, always supply them as a list. Now, you can run the experiment:

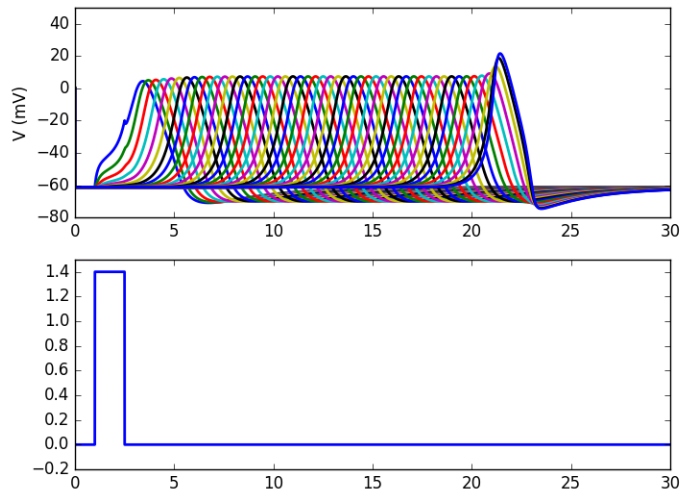
```
>>> xp.run(10, 0.002)
>>> plt.figure()
>>> plt.subplot(2,1,1)
>>> plt.plot(xp.T, soma.Vm, linewidth=2.0)
>>> plt.plot(xp.T, dend.Vm, linewidth=2.0)
>>> plt.ylim([-80,50])
>>> plt.subplot(2,1,2)
>>> plt.plot(xp.t, clp.Command, linewidth=2.0)
>>> plt.ylim([-1,2])
>>> plt.ylabel('V (mV)')
>>> plt.show()
```

You will see something like:



### Example 5: Try 50 Compartments

This will test how fast PyHH is and how easy to define 50 compartments. By now I think you can do it by yourself. See `ex5.py` for details, and You won't have any difficulty with these codes. You will get something like:



Do you know why the last compartment has the biggest action potential?

### Example 6. Voltage Clamp

We are going to build a single-compartment neuron to see how the voltage clamp works. First, we import pyhh and pylab, and 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 clammer 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])
```

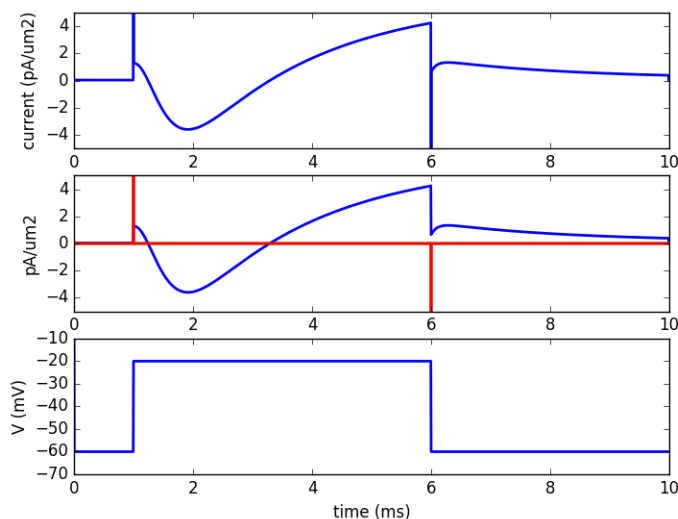
During real patch clamp experiments, we want to measure the transmembrane current ( $J_m$ ), and we always suppose that  $J_p = J_m$ . Actually even for ideal voltage clampers and ideal pipets,  $J_p$  is not equal to  $J_m$ .  $J_p$  is composed of at least two components, the transmembrane capacity current ( $J_c$ ) and the transmembrane ionic current ( $J_m$ ). If space clamp is not good, the cytosolic current ( $J_n$ ) also contributes to  $J_p$ . If  $J_n$  is small, we can expect  $J_p \approx J_m$ . In real experiment, we normally see  $J_p$ , not the components of  $J_p$ . The PyHH VClamper stores three currents in the form of current density. So we are going to show  $J_m$  and  $J_c$  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  $J_p$ , which is the sum of  $J_c$  and  $J_m$ .  $J_n$  is zero in this example. The middle panel shows  $J_m$  (in blue) and  $J_c$  (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

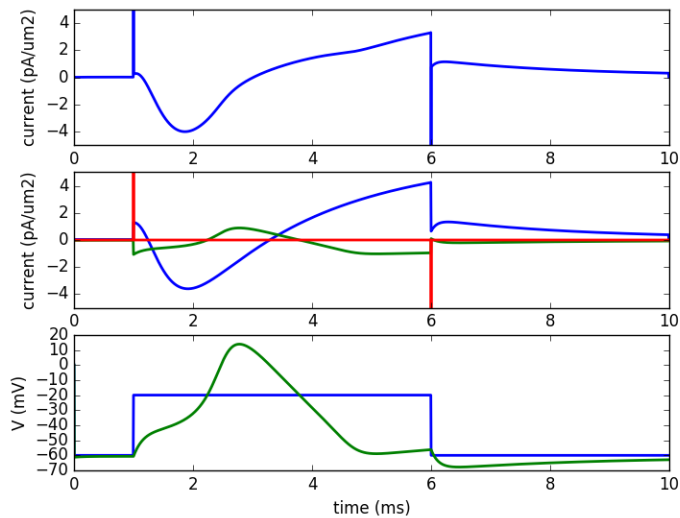
>>> soma = Compartment(diameter = 50, length=None)
>>> soma.add_channels([NaC, KDR, gL])
>>> dend = Compartment(diameter = 1.5, length = 100)
>>> dend.add_channels([NaC, KDR, gL])
>>> soma.connect(dend)

>>> 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,  $J_p$ . However, the blue curve in the middle channel is what we want ( $J_m$ ). We can see the difference between  $J_p$  and  $J_m$ , 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)
```

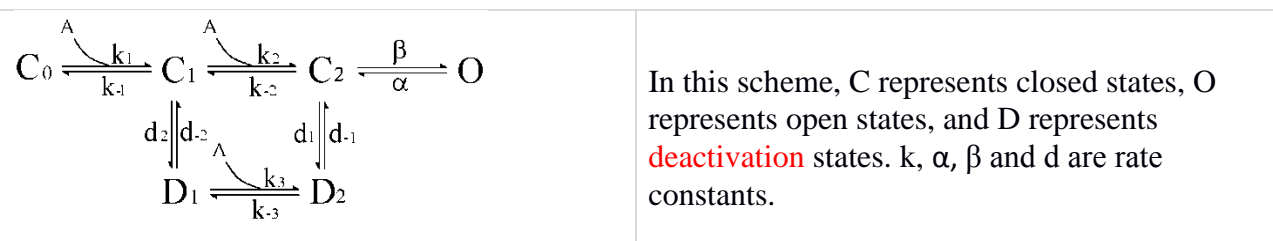
with

```
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.



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': 0.15, 'D2': 0.70},
    'D2': {'C2': 0.70, 'D1': 0.15},
    'O': {'C2': 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)

```

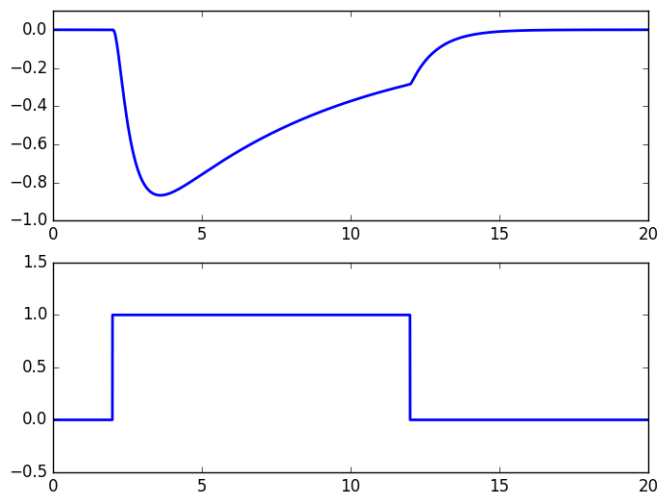
```

>>> xp.run(20,dt=0.005)

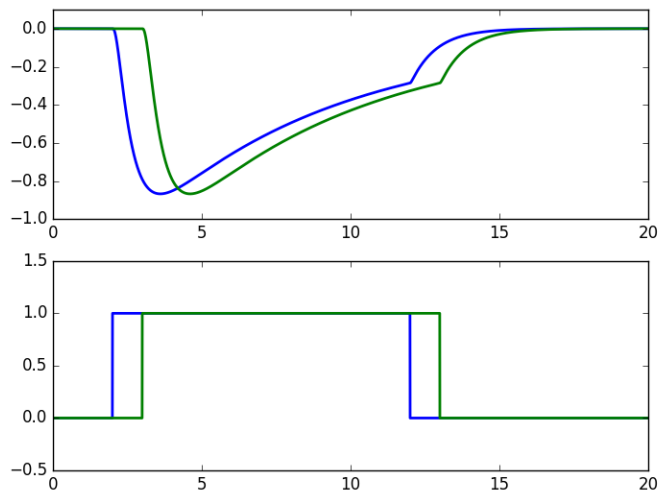
>>> plt.figure()
>>> plt.subplot(2,1,1)
>>> plt.plot(xp.T, vclp.Jm, linewidth=2.0)
>>> plt.ylim([-1,0.1])
>>> plt.subplot(2,1,2)
>>> plt.plot(xp.T, deliver.Command, linewidth=2.0)
>>> plt.ylim([-0.5,1.5])
>>> plt.show()

```

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.