Tutorial 1: the Basic

PyHH is a small library of Python for computational simulation of the electrical behavior 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). Do a simulation with these classes is just like performing a real experiment in a lab. I will start by some simple examples, then I will go through some details of the codes.

Installation:

You need Python, but you don't need installation of PyHH, just drop PyHH.py in a folder where you want to use it. However, you need something for visualization of the results, such as Matplotlib. Check on the Internet for installation of Python and Matplotlib.

Now let's start. In a terminal, cd into the folder where you keep PyHH, test if PyHH works by typing

```
$ python ex5.py
```

If you see something, it works. Otherwise, something is wrong and let me know so I can fix it.

Now you can go to the Python environment to experience PyHH step by step. Type

```
$ python
```

or

```
$ python3
```

I found PyHH runs faster with 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)
```

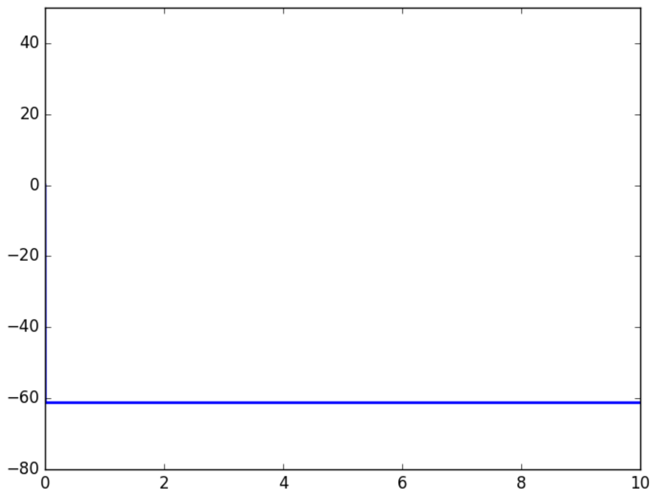
This defines an experiment, which gathers information from the compartment for integration of equations.

```
>>> 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 matlibplot:

```
>>> 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:

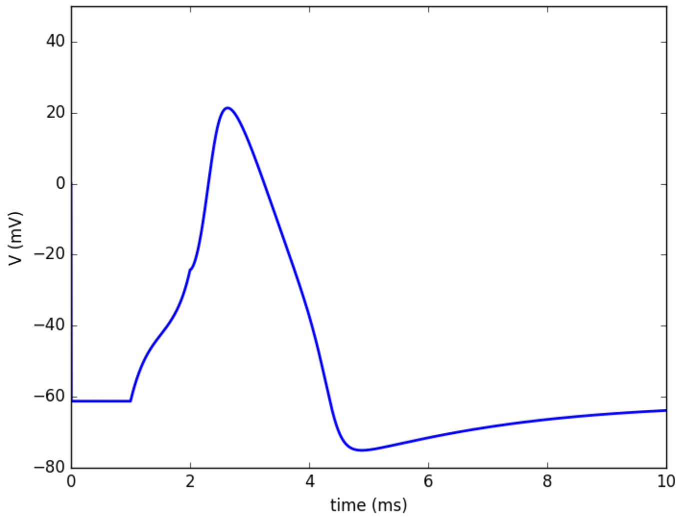
```
>>> clp.connect(cpm)
```

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 pA/μm^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)
```

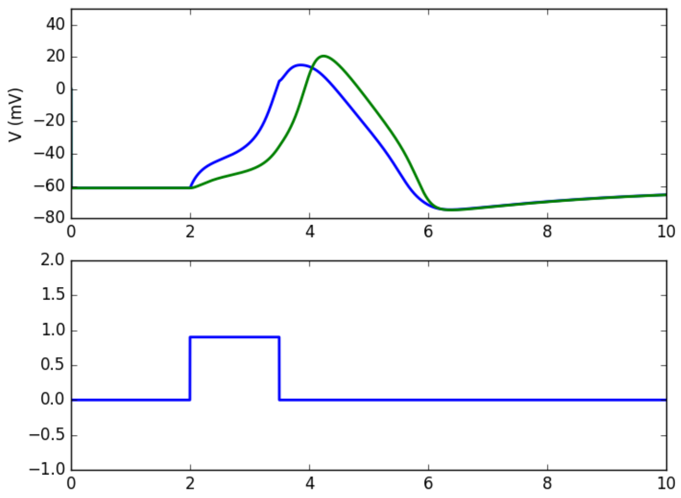
Here is the trick, you can not do dend.connect(soma). Connection is directed in PyHH. I define a function which perform the same task: dend.attached_to(soma)

```
>>> 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:

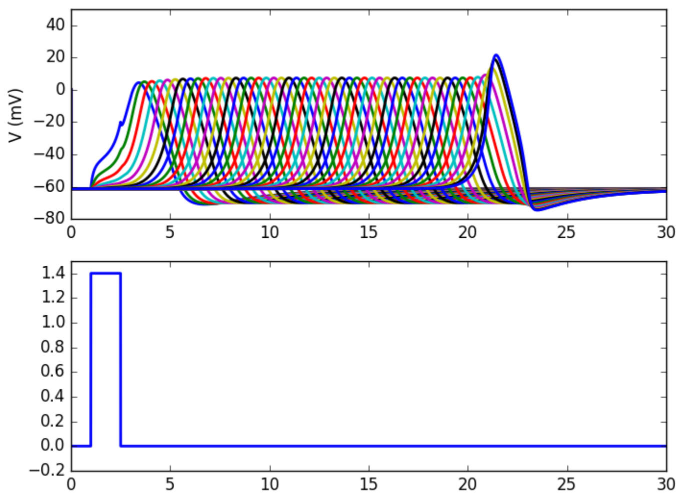
```
>>> xp.run(10, 0.002)
>>> plt.figure()
>>> plt.subplot(2,1,1)
>>> plt.plot(xp.T, soma.Vm, linewidth=2.0)
>>> plt.plot(xp.T, dend.Vm, linewidth=2.0)
>>> plt.ylim([-80,50])
>>> plt.subplot(2,1,2)
>>> plt.plot(xp.t, clp.Command, linewidth=2.0)
>>> plt.ylim([-1,2])
>>> plt.ylabel('V (mV)')
>>> plt.show()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

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?