Introduction to Neural Networks

Programming Lab 3: Spiking neuron models in Brian

Getting started

Today, we will install Brian2 and implement some simple single-neuron models. To start, we will install Brian2 by typing at the command prompt:

```
$ conda install -c conda-forge brian2
```

which will prompt the Anaconda distribution to search for the `brian2` package and begin installation. If you run into any problems, troubleshooting guides for package installation are provided by Anaconda. We can assist you with the installation process. After `brian2` has been successfully installed in your Anaconda distribution, verify that you can import the module at the `ipython` command prompt:

```
from brian2 import *
```

After verifying that the module imports without errors, we will then import our plotting module:

```
import matplotlib.pyplot as plt
```

and introduce the preliminaries for running simulations in Brian. There are detailed tutorials on the Brian website, and there is also a summary paper on using Brian for spiking neuron simulations.

Physical units

One of the most useful features of Brian is the incorporation of physical units, which allows us to use "millivolts" or "nanoamperes" directly in simulations. This simplifies our numerical calculations significantly, we don't have to keep track of factors such as 10^{-9} throughout our code, and we can instead work with the units as we would in manual calculations.

To start with this, we can define the membrane time constant τ_m :

```
tau = 20*ms
```

which will then define `tau` in our code with a value of 20 milliseconds. Other physical units available for our simulations are "volts", "amps", and "ohm". We can also use standard prefixes in the SI units system such as "mV", "nA", "ms", or "millisecond". The units system combines units naturally, so that the result of multiplying a current by a resistance is a potential:

```
1*nA * 5*Mohm
>>> 5*mV
```

Also, the unit system performs error checking, so that if we try to combine units incorrectly:

```
1*nA + 5*mV
```

we will receive an error. This becomes a very useful and important feature when implementing and debugging

equations describing neuron models in Brian.

A simple model

We will first implement a simple differential equation so that we can get started with defining, running, and plotting simulations in Brian. We will start by defining a parameter and the differential equation defining the model:

```
start_scope() # clears the workspace of previous Brian objects

# parameters and model equations
tau = 20*ms
eqs = '''
dx/dt = -x/tau : 1
'''

# create model
N = NeuronGroup( 1, eqs )

# initialize model
N.x = 0

# record model state
M = StateMonitor( N, 'x', record=True )
```

which will create a NeuronGroup with 1 model neuron. We can then run the simulation for 100 milliseconds:

```
run(100*ms)
```

and now, we will add increase the value of `x` to 1:

```
N.x = 1
```

and run the simulation for another 100 milliseconds:

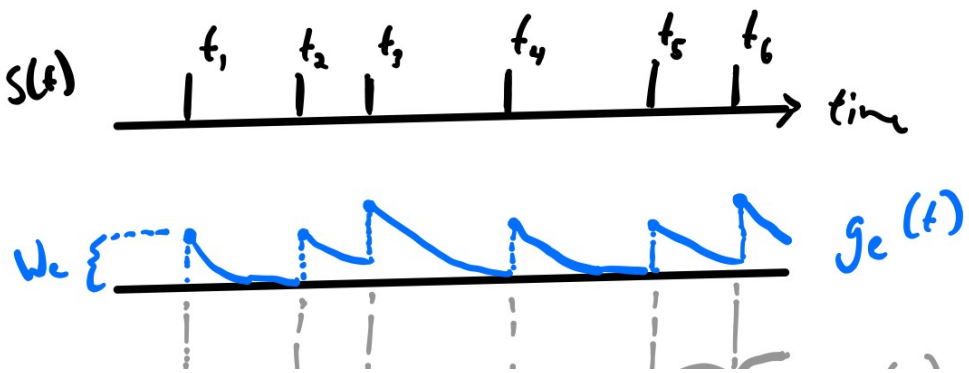
```
run(100*ms)
```

Finally, we will plot the result:

```
plt.plot( M.t/ms, M.x[0] )
```

which illustrates the simple decay dynamics resulting from the differential equation.

Simulation 1: LIF neuron with excitatory inputs



LIF Neuron with inhibitory Poisson inputs.

We can now simulate the LIF neuron receiving excitatory synaptic inputs:

$$\tau_m \frac{dV}{dt} = E_L - V + g_e(E_e - V)$$

$$\tau_e \frac{dg_e}{dt} = -g_e$$

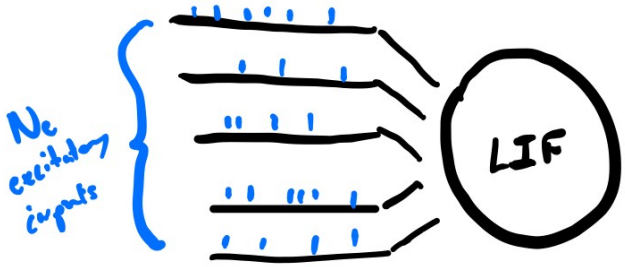
where we will specify the input spike times using the `SpikeGeneratorGroup` in Brian. The input spikes (arriving at 25, 50, and 75 milliseconds during the simulation) create jump increases in the synaptic conductance g_e by the synaptic weight w_e and depolarizing excitatory post-synaptic potentials (EPSPs) in the V_m trace.

```
start_scope()

# parameters
taum   = 20*ms    # time constant
g_L    = 10*nS    # leak conductance
E_l    = -70*mV   # leak reversal potential
E_e    = 0*mV     # excitatory reversal potential
tau_e  = 5*ms     # excitatory synaptic time constant
Vr     = E_l      # reset potential
Vth    = -50*mV    # spike threshold
Vs     = 20*mV    # spiking potential
w_e    = 1        # excitatory synaptic weight (units of g_L)

# model equations
eqs = '''
dv/dt = ( E_l - v + g_e*(E_e-v) ) / taum  : volt (unless refractory)
dg_e/dt = -g_e/tau_e  : 1  # excitatory conductance (dimensionless units)
'''

# create neuron
N = NeuronGroup( 1, model=eqs, threshold='v>Vth', reset='v=Vr', refractory='5*ms', met

# initialize neuron
N.v = E_l

# create inputs
indices = array([0, 0, 0]); times = array([25, 50, 75])*ms
input = SpikeGeneratorGroup( 1, indices, times )

# create connections
S = Synapses( input, N, 'w: 1', on_pre='g_e += w_e' )
S.connect( i=0, j=0 );

# record model state
M = StateMonitor( N, ('v','g_e'), record=True )
```

```
# run simulation
run( 100*ms )

# plot output
fig, ax1 = plt.subplots(); ax2 = ax1.twinx()
ax1.plot( M.t/ms, M.v[0] ); ax2.plot( M.t/ms, M.g_e[0], 'g--' );
ax1.set_xlabel( 'time (ms)' ); ax1.set_ylabel( 'V_m (V)' ); ax2.set_ylabel( 'g_e (unit
```

Experiment: How do the EPSPs change at different resting potentials? What effect could this have on neuronal integration of incoming inputs?

Simulation 2: LIF neuron with Poisson excitatory inputs



LIF Neuron with excitatory Poisson inputs.

Instead of specifying each spike time manually in our simulation, we will now use random synaptic inputs that are generated by a Poisson process (`PoissonGroup` in Brian). We will cover the details of the Poisson process in lecture this week. The model equations are the same as above:

$$\tau_m \frac{dV}{dt} = E_L - V + g_e(E_e - V)$$

$$\tau_e \frac{dg_e}{dt} = -g_e$$

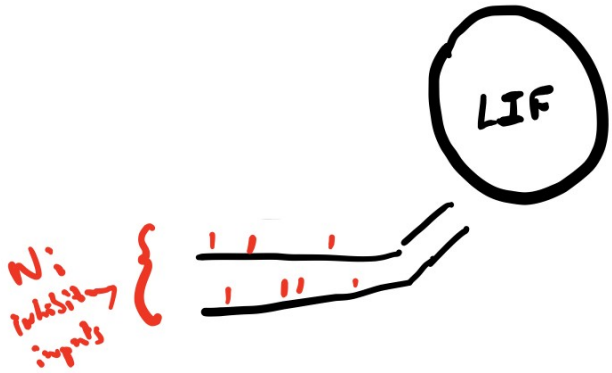
where the firing rate of the synaptic inputs is now specified by ν_e .

```
start_scope()

# parameters
taum   = 20*ms   # time constant
g_L    = 10*nS   # leak conductance
E_l    = -70*mV  # leak reversal potential
E_e    = 0*mV    # excitatory reversal potential
tau_e  = 5*ms    # excitatory synaptic time constant
Vr     = E_l     # reset potential
Vth    = -50*mV  # spike threshold
Vs     = 20*mV   # spiking potential
w_e    = 0.1     # excitatory synaptic weight (units of g_L)
v_e    = 5*Hz    # excitatory Poisson rate
N_e    = 100     # number of excitatory inputs

# model equations
eqs = '''
dv/dt = ( E_l - v + g_e*(E_e-v) ) / taum  : volt (unless refractory)
dg_e/dt = -g_e/tau_e  : 1  # excitatory conductance (dimensionless units)
```

```
'''

# create neuron
N = NeuronGroup( 1, model=eqs, threshold='v>Vth', reset='v=Vr', refractory='5*ms', met

# initialize neuron
N.v = E_l

# create inputs
P = PoissonGroup( N_e, v_e )

# create connections
S = Synapses( P, N, 'w: 1', on_pre='g_e += w_e' )
S.connect( i=0, j=0 );

# record model state
M = StateMonitor( N, ('v','g_e'), record=True )

# run simulation
run( 1000*ms )

# plot output
fig, ax1 = plt.subplots(); ax2 = ax1.twinx()
ax1.plot( M.t/ms, M.v[0] );
ax1.set_xlabel( 'time (ms)' ); ax1.set_ylabel( 'V_m (V)' ); ax2.set_ylabel( 'g_e (unit
```

Experiment: How do the membrane potential fluctuations change with different w_e and ν_e? What differences are there between the regime of w_e small and ν_e large versus w_e large and ν_e small?

Simulation 3: LIF neuron with Poisson inhibitory inputs



LIF Neuron with inhibitory Poisson inputs. (left)

Next we will add inhibitory synaptic inputs, following a Poisson process with rate ν_i:

$$\tau_m \frac{dV}{dt} = E_L - V + g_i(E_i - V)$$

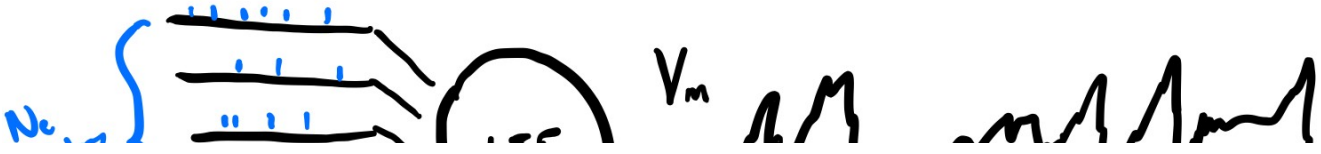
$$\tau_i \frac{dg_i}{dt} = -g_i$$

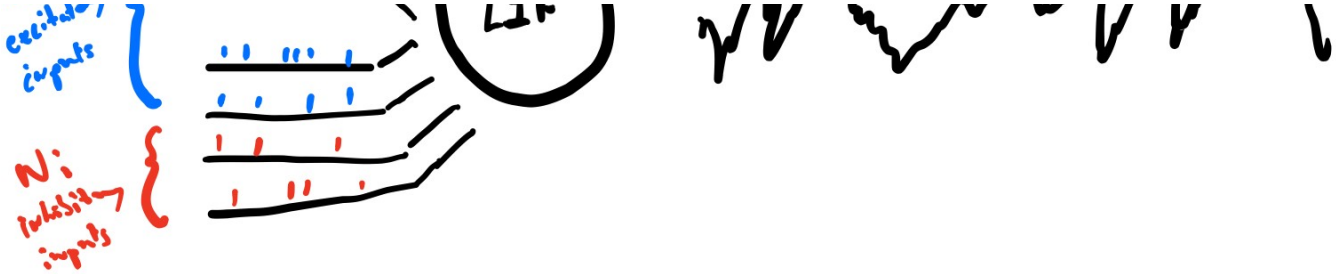
and the inputs will have the same synaptic weight, specified by the parameter w_i.

```
start_scope()

# parameters
taum   = 20*ms   # time constant
g_L    = 10*nS   # leak conductance
E_l    = -70*mV  # leak reversal potential
E_i    = -80*mV  # inhibitory reversal potential
tau_i  = 10*ms   # inhibitory synaptic time constant
Vr     = E_l     # reset potential
Vth    = -50*mV   # spike threshold
Vs     = 20*mV   # spiking potential
w_i    = 0.1     # inhibitory synaptic weight
v_i    = 5*Hz    # inhibitory Poisson rate
N_i    = 100     # number of inhibitory inputs

# model equations
eqs = '''
dv/dt = ( E_l - v + g_i*(E_i-v) ) / taum  : volt (unless refractory)
dg_i/dt = -g_i/tau_i  : 1  # excitatory conductance (dimensionless units)
'''

# create neuron
N = NeuronGroup( 1, model=eqs, threshold='v>Vth', reset='v=Vr', refractory='5*ms', met

# initialize neuron
N.v = E_l

# create inputs
P = PoissonGroup( N_i, v_i )

# create connections
S = Synapses( P, N, 'w: 1', on_pre='g_i += w' )
S.connect( p=1 );

# record model state
M = StateMonitor( N, ('v','g_i'), record=True )

# run simulation
run( 1000*ms )

# plot output
fig, ax1 = plt.subplots(); ax2 = ax1.twinx()
ax1.plot( M.t/ms, M.v[0] ); ax2.plot( M.t/ms, M.g_i[0], 'g--' );
ax1.set_xlabel( 'time (ms)' ); ax1.set_ylabel( 'V_m (V)' ); ax2.set_ylabel( 'g_i (unit
```

Experiment: Similar to the experiment above, how do the membrane potential fluctuations change with different w_i and ν_i? What differences are there between the regime of w_i small and ν_i large versus w_i large and ν_i small? How do the EPSPs change between a membrane potential of -70 mV and -55 mV?

Simulation 4: LIF neuron with E/I Poisson inputs





LIF Neuron with balanced excitatory and inhibitory Poisson inputs. (left) The LIF neuron model (circle) receives input from $N_{\{e,i\}}$ excitatory and inhibitory Poisson synaptic inputs (far left), each firing at a rate $\nu_{\{e,i\}}$ (blue and red spikes, respectively). **(right)** The resulting membrane potential trace V_m depends on the balance of excitatory and inhibitory drives created by these inputs.

Finally, we will simulate the full single-neuron model receiving balanced excitatory and inhibitory inputs:

$$\tau_m \frac{dV}{dt} = E_L - V + g_e(E_e - V) + g_i(E_i - V)$$

$$\tau_e \frac{dg_e}{dt} = -g_e$$

$$\tau_i \frac{dg_i}{dt} = -g_i$$

where we will consider N synaptic inputs, with $N_e = \lfloor 0.8N \rfloor$ excitatory inputs and $N_i = \lfloor 0.2N \rfloor$ inhibitory inputs, again reflecting that cortical networks are composed of 80% excitatory neurons and 20% inhibitory neurons. The synaptic weights $w_{\{e,i\}}$ and the input rates $\nu_{\{e,i\}}$ can be adjusted to create conditions with varying balance of excitation and inhibition.

```
start_scope()

# parameters
taum    = 20*ms          # time constant
g_L     = 10*nS          # leak conductance
E_l     = -70*mV         # leak reversal potential
E_e     = 0*mV           # excitatory reversal potential
tau_e   = 5*ms           # excitatory synaptic time constant
E_i     = -80*mV         # inhibitory reversal potential
tau_i   = 10*ms          # inhibitory synaptic time constant
Nin     = 1000           # number of synaptic inputs
Ne      = int(0.8*Nin)   # number of excitatory inputs
Ni      = int(0.2*Nin)   # number of inhibitory inputs
Vr      = E_l            # reset potential
Vth     = -50*mV         # spike threshold
Vs      = 20*mV          # spiking potential

w_e     = 0.1              # excitatory synaptic weight
w_i     = 0.4              # inhibitory synaptic weight

ve      = 10*Hz          # excitatory input rate
vi      = 10*Hz          # inhibitory input rate

# model equations
eqs = '''
dv/dt = ( E_l - v + g_e*(E_e-v) + g_i*(E_i-v) ) / taum  : volt (unless refractory)
dg_e/dt = -g_e/tau_e  : 1  # excitatory conductance (dimensionless units)
dg_i/dt = -g_i/tau_i  : 1  # inhibitory conductance (dimensionless units)
```

```
'''

# create neuron
N = NeuronGroup( 1, model=eqs, threshold='v>Vth', reset='v=Vr', refractory='5*ms', met

# initialize neuron
N.v = E_l

# create inputs
Pe = PoissonGroup( 1, (ve*Ne) ); Pi = PoissonGroup( 1, (vi*Ni) )

# create connections
synE = Synapses( Pe, N, 'w: 1', on_pre='g_e += w_e' ); synE.connect( p=1 );
synI = Synapses( Pi, N, 'w: 1', on_pre='g_i += w_i' ); synI.connect( p=1 );

# record model state
M = StateMonitor( N, ('v','g_i'), record=True )
S = SpikeMonitor( N )

# run simulation
run( 1000*ms )

# plot output
plt.figure(figsize=(15,5)); plt.plot( M.t/ms, M.v[0] );
```

Experiment: How do the statistics of the membrane potential (mean, SD) change at different values of $w_{\{e,i\}}$ and $\nu_{\{e,i\}}$? Can you study the output rate ν_o as a function of these control parameters? What changes can you observe? What difference emerges between conductance-based synapses (i.e. where the synaptic input terms are $g_{\{e,i\}}(E_{\{e,i\}} - V)$ and current-based synapses (i.e. where we model synaptic input as a fixed, time-varying current injection following a pre-synaptic spike, equivalent to changing the synaptic input terms to $g_{\{e,i\}}(E_{\{e,i\}} - E_L)$ so that the response no longer depends on the post-synaptic neuron's present state?