Report 4

Computational Neuroscience

Computer Assignment 4

Aref Afzali

610098014

In [1]:
```
%matplotlib notebook
import torch
import numpy as np
```

In [2]:
```
from cnsproject.network.neural_populations import LIFPopulation
from cnsproject.plotting.plotting import plotting
from cnsproject.utils import noise_function, step_noise_function, incremental_step_noise
_function
from cnsproject.network.monitors import Monitor
from cnsproject.network.connections import Connection, randomNormalConnect, fullyConnect
, randomUniformConnect
```

Global Variables

$time$ parameter shows how often (seconds*scale/dt) we want to run our neuron. dt means with what resolution ($scale$) we want our seconds move forward. Ther will be 3 population which two of them are excitatory an one of thrm is inhibitory. The number of neurons in each population considered 100 neurons.

In [3]:
```
time = 1000
scale = 100
dt = 1
neuron_size = 100
shape_ep1 = (int(neuron_size),)
shape_ep2 = (int(neuron_size),)
shape_ip1 = (int(neuron_size),)
```

Walkthrough

The default implemented model is a normal random connection and heterogeneous population.

First we need to specify the input current.

In [4]:
```
I_ep1 = incremental_step_noise_function(
        time = time, I_value = 20, scale = scale, neuron_size = shape_ep1[0], gap = 5
    )
I_ep2 = incremental_step_noise_function(
        time = time, I_value = 20, scale = scale, neuron_size = shape_ep2[0], gap = 3
    )
I_ip1 = step_noise_function(time = time, I_value = 20, scale = scale, neuron_size = shap
e_ip1[0])
```

Then the 3 population that mentioned before, are created by LIF Population model.

In [5]:
```
ep1 = LIFPopulation(
        shape = shape_ep1, spike_trace = True, additive_spike_trace = True,
        tau_s = 10, trace_scale = 1., is_inhibitory = False,
        learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ep2 = LIFPopulation(
        shape = shape_ep2, spike_trace = True, additive_spike_trace = True,
        tau_s = 10, trace_scale = 1., is_inhibitory = False,
        learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ip1 = LIFPopulation(
        shape = shape_ip1, spike_trace = True, additive_spike_trace = True,
        tau_s = 10, trace_scale = 1., is_inhibitory = True,
        learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
```

Here we creat connection between the populations. We have a connection from each excitatory population to the inhibitory population and vice versa. And each excitatory population has a connection with itself.

In [6]:
```
con_ep1_ip1 = Connection(
        pre = ep1, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ip1 = Connection(
        pre = ep2, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep1 = Connection(
        pre = ip1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep2 = Connection(
        pre = ip1, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep1_ep1 = Connection(
        pre = ep1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ep2 = Connection(
        pre = ep2, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
```

For each population we need to configure a monitor.

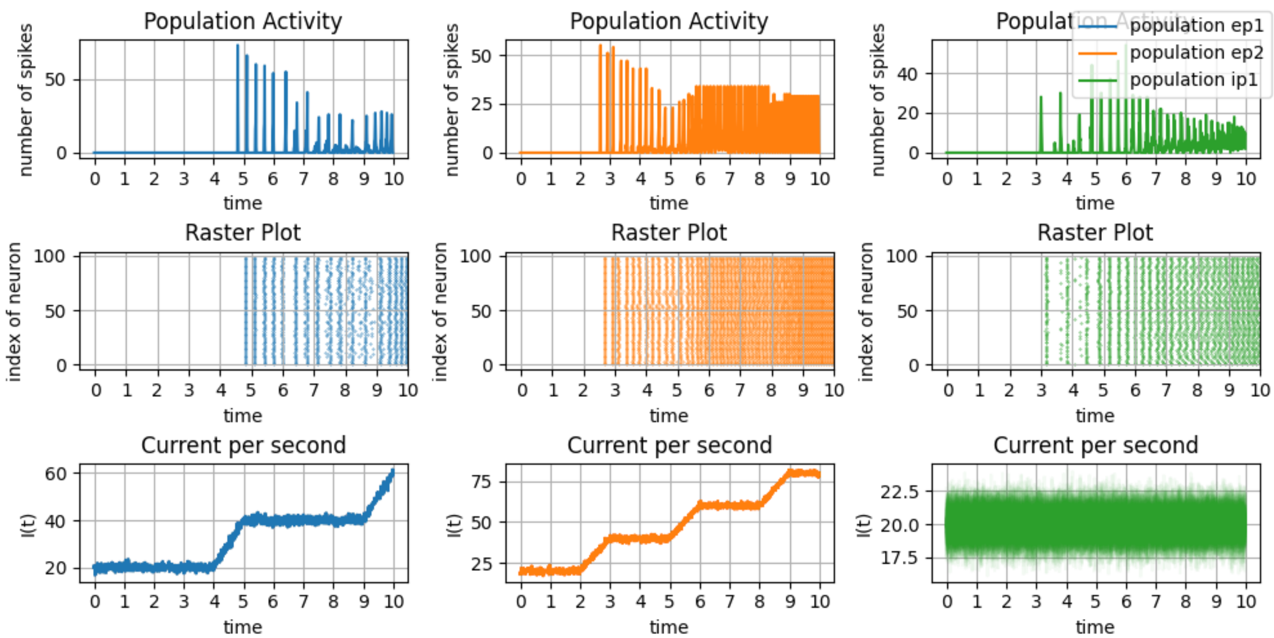
In [7]:
```
monitor_ep1 = Monitor(ep1, state_variables=["s", "u"])
monitor_ep1.set_time_steps(time, dt)
monitor_ep1.reset_state_variables()

monitor_ep2 = Monitor(ep2, state_variables=["s", "u"])
monitor_ep2.set_time_steps(time, dt)
monitor_ep2.reset_state_variables()

monitor_ip1 = Monitor(ip1, state_variables=["s", "u"])
monitor_ip1.set_time_steps(time, dt)
monitor_ip1.reset_state_variables()
```

At a time, each population will forward its current and then compute its effect in each connection.

In [8]:
```
out_ep1_ip1 = 0
out_ep2_ip1 = 0
out_ip1_ep1 = 0
out_ip1_ep2 = 0
out_ep1_ep1 = 0
out_ep2_ep2 = 0
for i in range(time):
    ep1.forward(I_ep1[i] - out_ip1_ep1 + out_ep1_ep1)
    ep2.forward(I_ep2[i] - out_ip1_ep2 + out_ep2_ep2)
    ip1.forward(I_ip1[i] + out_ep1_ip1 + out_ep2_ip1)
    out_ep1_ip1 = con_ep1_ip1.compute()
    out_ep2_ip1 = con_ep2_ip1.compute()
    out_ip1_ep1 = con_ip1_ep1.compute()
    out_ip1_ep2 = con_ip1_ep2.compute()
    out_ep1_ep1 = con_ep1_ep1.compute()
    out_ep2_ep2 = con_ep2_ep2.compute()
    monitor_ep1.record()
    monitor_ep2.record()
    monitor_ip1.record()

s_ep1 = torch.transpose(monitor_ep1.get("s")*1, 0, 1)
s_ep2 = torch.transpose(monitor_ep2.get("s")*1, 0, 1)
s_ip1 = torch.transpose(monitor_ip1.get("s")*1, 0, 1)
```

In the end lets plot the output.

In [9]:
```
plot = plotting()

plot.plot_three_population_activity_init(time/scale)
plot.plot_three_population_activity_update(s_ep1, I_ep1, s_ep2, I_ep2, s_ip1, I_ip1, n1=
"ep1", n2="ep2", n3="ip1")
plot.show()
```



Population Behavior

Zero Current to Inhibitory Population

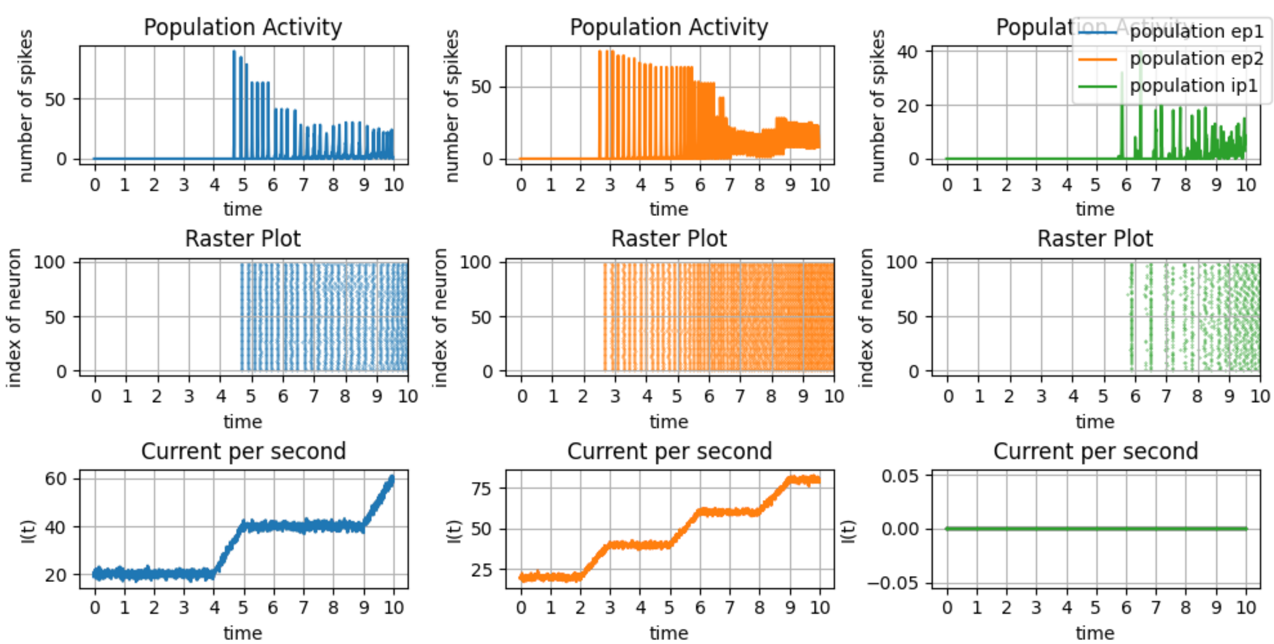
If we change the input of the inhibitory population to zero, we can see the effect of the other excitatory populations on it and they cause it to spike.

In [10]:

```
time = 1000
neuron_size = 100
shape_ep1 = (int(neuron_size),)
shape_ep2 = (int(neuron_size),)
shape_ip1 = (int(neuron_size),)

I_ep1 = incremental_step_noise_function(time = time, I_value = 20, scale = scale, neuron
_size = shape_ep1[0], gap = 5)
I_ep2 = incremental_step_noise_function(time = time, I_value = 20, scale = scale, neuron
_size = shape_ep2[0], gap = 3)
I_ip1 = torch.zeros(time, neuron_size)

ep1 = LIFPopulation(
        shape = shape_ep1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ep2 = LIFPopulation(
        shape = shape_ep2, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
```

```
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ip1 = LIFPopulation(
        shape = shape_ip1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = True, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )

con_ep1_ip1 = Connection(
        pre = ep1, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ip1 = Connection(
        pre = ep2, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep1 = Connection(
        pre = ip1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep2 = Connection(
        pre = ip1, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep1_ep1 = Connection(
        pre = ep1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ep2 = Connection(
        pre = ep2, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )


monitor_ep1 = Monitor(ep1, state_variables=["s", "u"])
monitor_ep1.set_time_steps(time, dt)
monitor_ep1.reset_state_variables()

monitor_ep2 = Monitor(ep2, state_variables=["s", "u"])
monitor_ep2.set_time_steps(time, dt)
monitor_ep2.reset_state_variables()

monitor_ip1 = Monitor(ip1, state_variables=["s", "u"])
monitor_ip1.set_time_steps(time, dt)
monitor_ip1.reset_state_variables()

out_ep1_ip1 = 0
out_ep2_ip1 = 0
out_ip1_ep1 = 0
out_ip1_ep2 = 0
out_ep1_ep1 = 0
out_ep2_ep2 = 0
for i in range(time):
```

```
    ep1.forward(I_ep1[i] - out_ip1_ep1 + out_ep1_ep1)
    ep2.forward(I_ep2[i] - out_ip1_ep2 + out_ep2_ep2)
    ip1.forward(I_ip1[i] + out_ep1_ip1 + out_ep2_ip1)
    out_ep1_ip1 = con_ep1_ip1.compute()
    out_ep2_ip1 = con_ep2_ip1.compute()
    out_ip1_ep1 = con_ip1_ep1.compute()
    out_ip1_ep2 = con_ip1_ep2.compute()
    out_ep1_ep1 = con_ep1_ep1.compute()
    out_ep2_ep2 = con_ep2_ep2.compute()
    monitor_ep1.record()
    monitor_ep2.record()
    monitor_ip1.record()

s_ep1 = torch.transpose(monitor_ep1.get("s")*1, 0, 1)
s_ep2 = torch.transpose(monitor_ep2.get("s")*1, 0, 1)
s_ip1 = torch.transpose(monitor_ip1.get("s")*1, 0, 1)

plot = plotting()

plot.plot_three_population_activity_init(time/scale)
plot.plot_three_population_activity_update(s_ep1, I_ep1, s_ep2, I_ep2, s_ip1, I_ip1, n1=
"ep1", n2="ep2", n3="ip1")
plot.show()
```



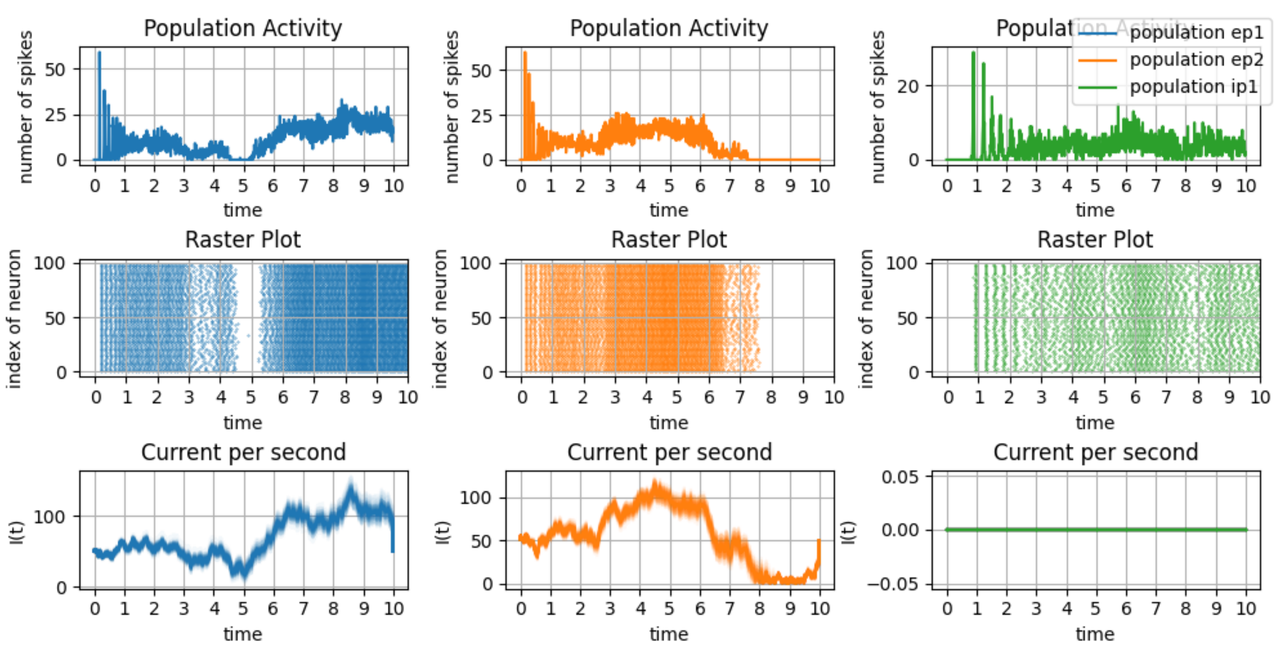
Random Noise as Input

Lets change the input current from a step random current to a random current. It seems that for a period of time we decide based on the first population and for another time we can decide based on the second population.

In [12]:
```
time = 1000
neuron_size = 100
shape_ep1 = (int(neuron_size),)
shape_ep2 = (int(neuron_size),)
shape_ip1 = (int(neuron_size),)
```

```
I_ep1 = noise_function(time = time, neuron_size = shape_ep1[0])
I_ep2 = noise_function(time = time, neuron_size = shape_ep2[0])
I_ip1 = torch.zeros(time, neuron_size)

ep1 = LIFPopulation(
        shape = shape_ep1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ep2 = LIFPopulation(
        shape = shape_ep2, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ip1 = LIFPopulation(
        shape = shape_ip1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = True, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )

con_ep1_ip1 = Connection(
        pre = ep1, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ip1 = Connection(
        pre = ep2, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep1 = Connection(
        pre = ip1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep2 = Connection(
        pre = ip1, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep1_ep1 = Connection(
        pre = ep1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ep2 = Connection(
        pre = ep2, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )


monitor_ep1 = Monitor(ep1, state_variables=["s", "u"])
monitor_ep1.set_time_steps(time, dt)
monitor_ep1.reset_state_variables()

monitor_ep2 = Monitor(ep2, state_variables=["s", "u"])
monitor_ep2.set_time_steps(time, dt)
monitor_ep2.reset_state_variables()
```

```
monitor_ip1 = Monitor(ip1, state_variables=["s", "u"])
monitor_ip1.set_time_steps(time, dt)
monitor_ip1.reset_state_variables()

out_ep1_ip1 = 0
out_ep2_ip1 = 0
out_ip1_ep1 = 0
out_ip1_ep2 = 0
out_ep1_ep1 = 0
out_ep2_ep2 = 0
for i in range(time):
    ep1.forward(I_ep1[i] - out_ip1_ep1 + out_ep1_ep1)
    ep2.forward(I_ep2[i] - out_ip1_ep2 + out_ep2_ep2)
    ip1.forward(I_ip1[i] + out_ep1_ip1 + out_ep2_ip1)
    out_ep1_ip1 = con_ep1_ip1.compute()
    out_ep2_ip1 = con_ep2_ip1.compute()
    out_ip1_ep1 = con_ip1_ep1.compute()
    out_ip1_ep2 = con_ip1_ep2.compute()
    out_ep1_ep1 = con_ep1_ep1.compute()
    out_ep2_ep2 = con_ep2_ep2.compute()
    monitor_ep1.record()
    monitor_ep2.record()
    monitor_ip1.record()

s_ep1 = torch.transpose(monitor_ep1.get("s")*1, 0, 1)
s_ep2 = torch.transpose(monitor_ep2.get("s")*1, 0, 1)
s_ip1 = torch.transpose(monitor_ip1.get("s")*1, 0, 1)

plot = plotting()

plot.plot_three_population_activity_init(time/scale)
plot.plot_three_population_activity_update(s_ep1, I_ep1, s_ep2, I_ep2, s_ip1, I_ip1, n1=
"ep1", n2="ep2", n3="ip1")
plot.show()
```

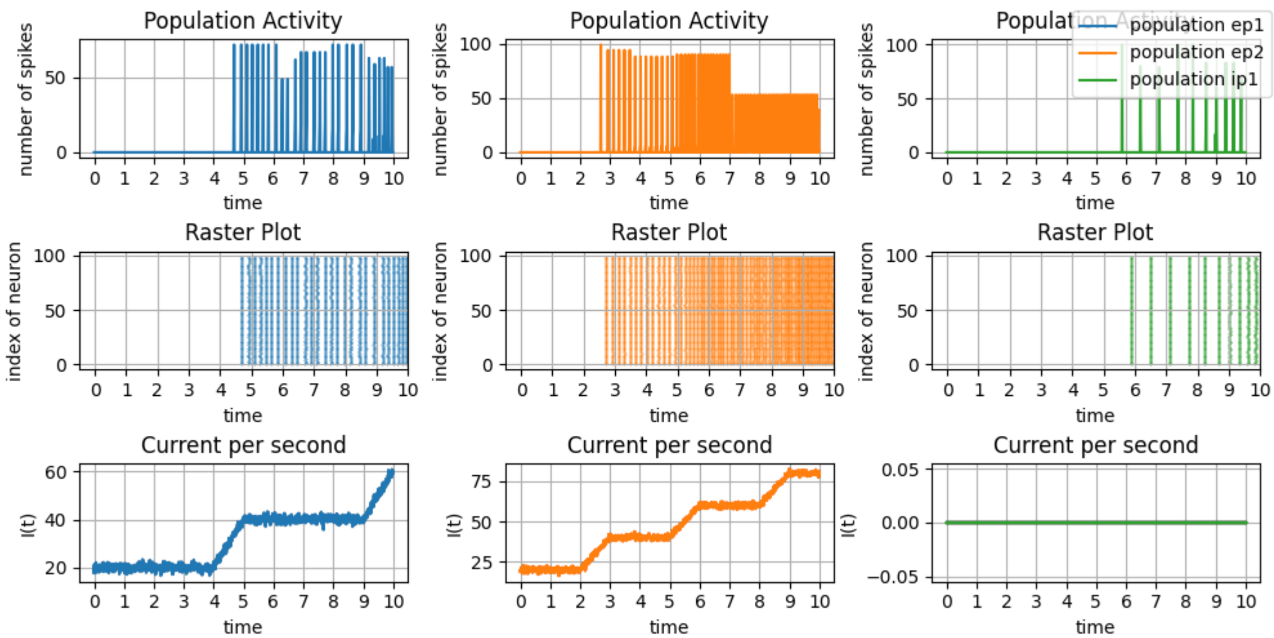


Uniform Random Connection

The next simulation has random step noise but the difference is instead of normal random connection, I used uniform random connection. It cause a more uniform activity for each population and cause for neurons spikes toghether.

In [17]:
```
time = 1000
neuron_size = 100
shape_ep1 = (int(neuron_size),)
shape_ep2 = (int(neuron_size),)
shape_ip1 = (int(neuron_size),)

I_ep1 = incremental_step_noise_function(time = time, I_value = 20, scale = scale, neuron
_size = shape_ep1[0], gap = 5)
I_ep2 = incremental_step_noise_function(time = time, I_value = 20, scale = scale, neuron
_size = shape_ep2[0], gap = 3)
I_ip1 = torch.zeros(time, neuron_size)

ep1 = LIFPopulation(
        shape = shape_ep1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ep2 = LIFPopulation(
        shape = shape_ep2, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ip1 = LIFPopulation(
        shape = shape_ip1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = True, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )

con_ep1_ip1 = Connection(
        pre = ep1, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomUniformConnect
, wmax=20., wmind=5.
    )
con_ep2_ip1 = Connection(
        pre = ep2, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomUniformConnect
, wmax=20., wmin=5.
    )
con_ip1_ep1 = Connection(
        pre = ip1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomUniformConnect
, wmax=20., wmin=5.
    )
con_ip1_ep2 = Connection(
        pre = ip1, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomUniformConnect
, wmax=20., wmin=5.
    )
con_ep1_ep1 = Connection(
        pre = ep1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomUniformConnect
, wmax=20., wmin=5.
    )
```

```
con_ep2_ep2 = Connection(
        pre = ep2, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomUniformConnect
, wmax=20., wmin=5.
    )


monitor_ep1 = Monitor(ep1, state_variables=["s", "u"])
monitor_ep1.set_time_steps(time, dt)
monitor_ep1.reset_state_variables()

monitor_ep2 = Monitor(ep2, state_variables=["s", "u"])
monitor_ep2.set_time_steps(time, dt)
monitor_ep2.reset_state_variables()

monitor_ip1 = Monitor(ip1, state_variables=["s", "u"])
monitor_ip1.set_time_steps(time, dt)
monitor_ip1.reset_state_variables()

out_ep1_ip1 = 0
out_ep2_ip1 = 0
out_ip1_ep1 = 0
out_ip1_ep2 = 0
out_ep1_ep1 = 0
out_ep2_ep2 = 0
for i in range(time):
    ep1.forward(I_ep1[i] - out_ip1_ep1 + out_ep1_ep1)
    ep2.forward(I_ep2[i] - out_ip1_ep2 + out_ep2_ep2)
    ip1.forward(I_ip1[i] + out_ep1_ip1 + out_ep2_ip1)
    out_ep1_ip1 = con_ep1_ip1.compute()
    out_ep2_ip1 = con_ep2_ip1.compute()
    out_ip1_ep1 = con_ip1_ep1.compute()
    out_ip1_ep2 = con_ip1_ep2.compute()
    out_ep1_ep1 = con_ep1_ep1.compute()
    out_ep2_ep2 = con_ep2_ep2.compute()
    monitor_ep1.record()
    monitor_ep2.record()
    monitor_ip1.record()

s_ep1 = torch.transpose(monitor_ep1.get("s")*1, 0, 1)
s_ep2 = torch.transpose(monitor_ep2.get("s")*1, 0, 1)
s_ip1 = torch.transpose(monitor_ip1.get("s")*1, 0, 1)

plot = plotting()

plot.plot_three_population_activity_init(time/scale)
plot.plot_three_population_activity_update(s_ep1, I_ep1, s_ep2, I_ep2, s_ip1, I_ip1, n1=
"ep1", n2="ep2", n3="ip1")
plot.show()
```



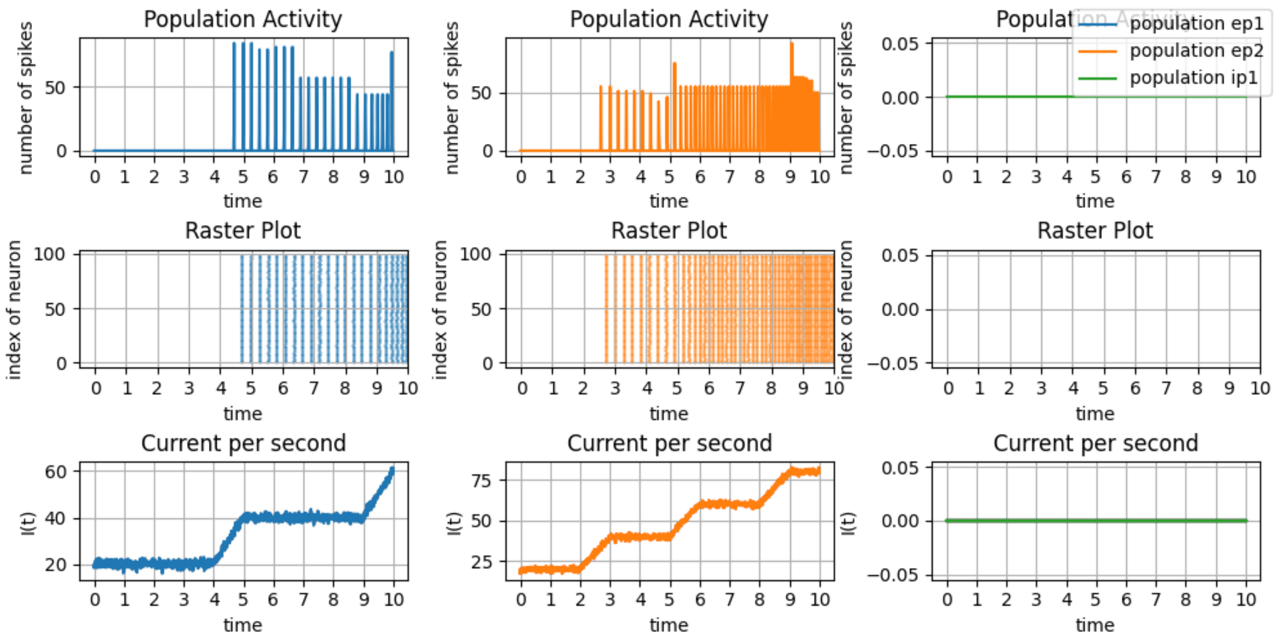
Fully Connected model

The next simulation has random step noise but the difference is instead of normal random connection, I used fully connection. It has less effect on the other population because its output will distribute among all the other population's neuron so it will cause less or no spikes for a inhibitory population without input current.

In [15]:
```
time = 1000
neuron_size = 100
shape_ep1 = (int(neuron_size),)
shape_ep2 = (int(neuron_size),)
shape_ip1 = (int(neuron_size),)

I_ep1 = incremental_step_noise_function(time = time, I_value = 20, scale = scale, neuron
_size = shape_ep1[0], gap = 5)
I_ep2 = incremental_step_noise_function(time = time, I_value = 20, scale = scale, neuron
_size = shape_ep2[0], gap = 3)
I_ip1 = torch.zeros(time, neuron_size)

ep1 = LIFPopulation(
        shape = shape_ep1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ep2 = LIFPopulation(
        shape = shape_ep2, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ip1 = LIFPopulation(
        shape = shape_ip1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = True, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )

con_ep1_ip1 = Connection(
        pre = ep1, post = ip1, lr = None, weight_decay = 0.0,
```

```
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= fullyConnect
    )
con_ep2_ip1 = Connection(
        pre = ep2, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= fullyConnect
    )
con_ip1_ep1 = Connection(
        pre = ip1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= fullyConnect
    )
con_ip1_ep2 = Connection(
        pre = ip1, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= fullyConnect
    )
con_ep1_ep1 = Connection(
        pre = ep1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= fullyConnect
    )
con_ep2_ep2 = Connection(
        pre = ep2, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= fullyConnect
    )


monitor_ep1 = Monitor(ep1, state_variables=["s", "u"])
monitor_ep1.set_time_steps(time, dt)
monitor_ep1.reset_state_variables()

monitor_ep2 = Monitor(ep2, state_variables=["s", "u"])
monitor_ep2.set_time_steps(time, dt)
monitor_ep2.reset_state_variables()

monitor_ip1 = Monitor(ip1, state_variables=["s", "u"])
monitor_ip1.set_time_steps(time, dt)
monitor_ip1.reset_state_variables()

out_ep1_ip1 = 0
out_ep2_ip1 = 0
out_ip1_ep1 = 0
out_ip1_ep2 = 0
out_ep1_ep1 = 0
out_ep2_ep2 = 0
for i in range(time):
    ep1.forward(I_ep1[i] - out_ip1_ep1 + out_ep1_ep1)
    ep2.forward(I_ep2[i] - out_ip1_ep2 + out_ep2_ep2)
    ip1.forward(I_ip1[i] + out_ep1_ip1 + out_ep2_ip1)
    out_ep1_ip1 = con_ep1_ip1.compute()
    out_ep2_ip1 = con_ep2_ip1.compute()
    out_ip1_ep1 = con_ip1_ep1.compute()
    out_ip1_ep2 = con_ip1_ep2.compute()
    out_ep1_ep1 = con_ep1_ep1.compute()
    out_ep2_ep2 = con_ep2_ep2.compute()
    monitor_ep1.record()
    monitor_ep2.record()
    monitor_ip1.record()

s_ep1 = torch.transpose(monitor_ep1.get("s")*1, 0, 1)
s_ep2 = torch.transpose(monitor_ep2.get("s")*1, 0, 1)
s_ip1 = torch.transpose(monitor_ip1.get("s")*1, 0, 1)
```

```
plot = plotting()

plot.plot_three_population_activity_init(time/scale)
plot.plot_three_population_activity_update(s_ep1, I_ep1, s_ep2, I_ep2, s_ip1, I_ip1, n1=
"ep1", n2="ep2", n3="ip1")
plot.show()
```



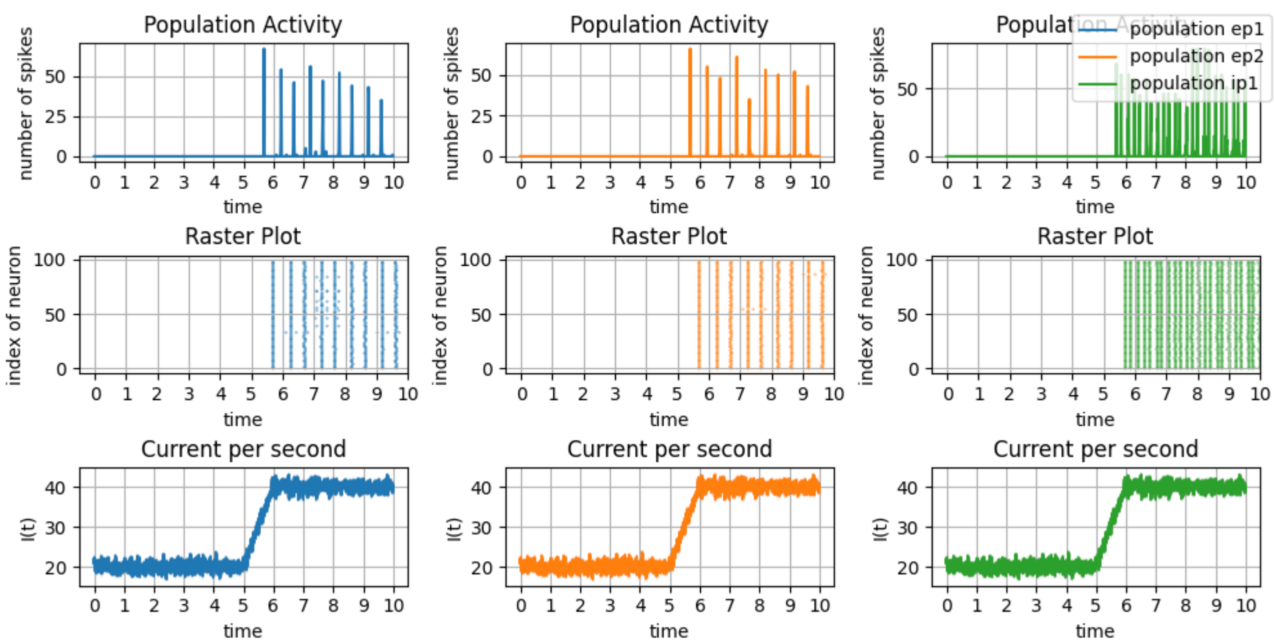
Same Current as Input

If we get a same current to all the populations, the result is as follows:

In [23]:
```
time = 1000
neuron_size = 100
shape_ep1 = (int(neuron_size),)
shape_ep2 = (int(neuron_size),)
shape_ip1 = (int(neuron_size),)

I_ep1 = incremental_step_noise_function(time = time, I_value = 20, scale = scale, neuron
_size = shape_ep1[0], gap = 6)
I_ep2 = torch.clone(I_ep1)
I_ip1 = torch.clone(I_ep1)

ep1 = LIFPopulation(
        shape = shape_ep1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ep2 = LIFPopulation(
        shape = shape_ep2, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ip1 = LIFPopulation(
        shape = shape_ip1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
```

```
        is_inhibitory = True, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )

con_ep1_ip1 = Connection(
        pre = ep1, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ip1 = Connection(
        pre = ep2, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep1 = Connection(
        pre = ip1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep2 = Connection(
        pre = ip1, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep1_ep1 = Connection(
        pre = ep1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ep2 = Connection(
        pre = ep2, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )


monitor_ep1 = Monitor(ep1, state_variables=["s", "u"])
monitor_ep1.set_time_steps(time, dt)
monitor_ep1.reset_state_variables()

monitor_ep2 = Monitor(ep2, state_variables=["s", "u"])
monitor_ep2.set_time_steps(time, dt)
monitor_ep2.reset_state_variables()

monitor_ip1 = Monitor(ip1, state_variables=["s", "u"])
monitor_ip1.set_time_steps(time, dt)
monitor_ip1.reset_state_variables()

out_ep1_ip1 = 0
out_ep2_ip1 = 0
out_ip1_ep1 = 0
out_ip1_ep2 = 0
out_ep1_ep1 = 0
out_ep2_ep2 = 0
for i in range(time):
    ep1.forward(I_ep1[i] - out_ip1_ep1 + out_ep1_ep1)
    ep2.forward(I_ep2[i] - out_ip1_ep2 + out_ep2_ep2)
    ip1.forward(I_ip1[i] + out_ep1_ip1 + out_ep2_ip1)
    out_ep1_ip1 = con_ep1_ip1.compute()
    out_ep2_ip1 = con_ep2_ip1.compute()
```

```
    out_ip1_ep1 = con_ip1_ep1.compute()
    out_ip1_ep2 = con_ip1_ep2.compute()
    out_ep1_ep1 = con_ep1_ep1.compute()
    out_ep2_ep2 = con_ep2_ep2.compute()
    monitor_ep1.record()
    monitor_ep2.record()
    monitor_ip1.record()

s_ep1 = torch.transpose(monitor_ep1.get("s")*1, 0, 1)
s_ep2 = torch.transpose(monitor_ep2.get("s")*1, 0, 1)
s_ip1 = torch.transpose(monitor_ip1.get("s")*1, 0, 1)

plot = plotting()

plot.plot_three_population_activity_init(time/scale)
plot.plot_three_population_activity_update(s_ep1, I_ep1, s_ep2, I_ep2, s_ip1, I_ip1, n1=
"ep1", n2="ep2", n3="ip1")
plot.show()
```



One Strong Current

As we can see below there may be a time when the inhibitory population won't allow another population spike although in the previous parts it spikes.

In [24]:
```
time = 1000
neuron_size = 100
shape_ep1 = (int(neuron_size),)
shape_ep2 = (int(neuron_size),)
shape_ip1 = (int(neuron_size),)

I_ep1 = incremental_step_noise_function(time = time, I_value = 20, scale = scale, neuron
_size = shape_ep1[0], gap = 6)
I_ep2 = torch.clone(I_ep1)
I_ep2[5*scale:,:] = I_ep2[5*scale:,:] + 20
I_ip1 = I_ep1
```

```
ep1 = LIFPopulation(
        shape = shape_ep1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ep2 = LIFPopulation(
        shape = shape_ep2, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = False, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )
ip1 = LIFPopulation(
        shape = shape_ip1, spike_trace = True, additive_spike_trace = True, tau_s = 10,
trace_scale = 1.,
        is_inhibitory = True, learning = False, R = 1, C = 20, threshold = -40, dt = dt
    )

con_ep1_ip1 = Connection(
        pre = ep1, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ip1 = Connection(
        pre = ep2, post = ip1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep1 = Connection(
        pre = ip1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ip1_ep2 = Connection(
        pre = ip1, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep1_ep1 = Connection(
        pre = ep1, post = ep1, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )
con_ep2_ep2 = Connection(
        pre = ep2, post = ep2, lr = None, weight_decay = 0.0,
        J = 1, tau_s = 10, trace_scale = 1., dt = dt, connectivity= randomNormalConnect,
wmean=20., wstd=5.
    )


monitor_ep1 = Monitor(ep1, state_variables=["s", "u"])
monitor_ep1.set_time_steps(time, dt)
monitor_ep1.reset_state_variables()

monitor_ep2 = Monitor(ep2, state_variables=["s", "u"])
monitor_ep2.set_time_steps(time, dt)
monitor_ep2.reset_state_variables()

monitor_ip1 = Monitor(ip1, state_variables=["s", "u"])
monitor_ip1.set_time_steps(time, dt)
```

```
monitor_ip1.reset_state_variables()

out_ep1_ip1 = 0
out_ep2_ip1 = 0
out_ip1_ep1 = 0
out_ip1_ep2 = 0
out_ep1_ep1 = 0
out_ep2_ep2 = 0
for i in range(time):
    ep1.forward(I_ep1[i] - out_ip1_ep1 + out_ep1_ep1)
    ep2.forward(I_ep2[i] - out_ip1_ep2 + out_ep2_ep2)
    ip1.forward(I_ip1[i] + out_ep1_ip1 + out_ep2_ip1)
    out_ep1_ip1 = con_ep1_ip1.compute()
    out_ep2_ip1 = con_ep2_ip1.compute()
    out_ip1_ep1 = con_ip1_ep1.compute()
    out_ip1_ep2 = con_ip1_ep2.compute()
    out_ep1_ep1 = con_ep1_ep1.compute()
    out_ep2_ep2 = con_ep2_ep2.compute()
    monitor_ep1.record()
    monitor_ep2.record()
    monitor_ip1.record()

s_ep1 = torch.transpose(monitor_ep1.get("s")*1, 0, 1)
s_ep2 = torch.transpose(monitor_ep2.get("s")*1, 0, 1)
s_ip1 = torch.transpose(monitor_ip1.get("s")*1, 0, 1)

plot = plotting()

plot.plot_three_population_activity_init(time/scale)
plot.plot_three_population_activity_update(s_ep1, I_ep1, s_ep2, I_ep2, s_ip1, I_ip1, n1=
"ep1", n2="ep2", n3="ip1")
plot.show()
```