```python
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 12 10:35:10 2022

@author: Benjamin
"""

#%% Assignment 4 - Q1C
import numpy as np
import scipy.signal as sig
import matplotlib.pyplot as plt
import pandas as pd
np.random.seed(123)

def simulatePoe(a0, b0, s_a, t1, s_b, t2, a_b, t3, s_ab):
    # input:
        # a0, b0 = baseline firing rate hz
        # s_a, t1 = stim effect on A and lag
        # s_b, t2 = stim effect on B and lag
        # a_b, t3 = A baseline effect on B and lag
        # a_ab = stim effect on A effect on B
    # output:
        # 500000 long binary vec for stim on each trial
        # 500000 long spike trains for A and B

    # hardcoded parameters
    n = 500000
    rfp = 3
    stim_n = 500
    stim_dur = 500

    # generate stim vec
    stim = np.zeros(n+t1+t2+t3)
    stim_onset = np.random.randint(0,n/stim_n-stim_dur,stim_n) + np.arange(0,n,n/st
    for i in stim_onset.astype(int):
        stim[i:i+stim_dur] = 1

    # generate neuron A spike train
    a = np.zeros(n+t1+t2+t3)
    for i in range(t1+t2, n+t1+t2+t3):
        if not np.any(a[i-rfp:i]): # not in rf period
            p = a0 + s_a*stim[i-t1] + np.random.normal(0, 1)
            a[i] = np.random.binomial(1, p/1000)

    # generate neuron B spike train
    b = np.zeros(n+t1+t2+t3)
    for i in range(t1+t2+t3, n+t1+t2+t3):
```

```
        if not np.any(b[i-rfp:i]): # not in rf period
            p = b0 + s_b*stim[i-t2] + a_b*a[i-t3] + s_ab*stim[i-t2]*a[i-t3] + np.ra
            b[i] = np.random.binomial(1, p/1000)

    return np.array([b[t1+t2+t3:], a[t1+t2:-t3], stim[t1+t3:-t2], stim[t2:-t3-t1]])

#%% fit

import statsmodels.api as sm
import statsmodels.formula.api as smf
import numpy as np
import pandas as pd

# set params
a0 = 60 # A baseline rate Hz
b0 = 50 # B baseline rate Hz
s_a = 30 # stim increase A rate Hz
t1 = 1 # lag from stim to increase A in ms
s_b = 50 # stim increase B rate Hz
t2 = 3 # lag from stim to increase B in ms
a_b = 20 # A spike increase B rate in Hz
t3 = 10 # lag from A to increase B in ms
s_ab = 50 # additional stim increase A spike increase on B

# simulate data
sim_dt = simulatePoe(a0, b0, s_a, t1, s_b, t2, a_b, t3, s_ab)

# make df for ease of use
df = pd.DataFrame(data=sim_dt, columns=['B_t', 'A', 'S', 'S_a'])

# fit logistic model
res1 = smf.logit(formula='B_t ~ S + A + S_a:A', data=df).fit()
print(res1.summary())

# evaluate model predictions on rate to check it
g = np.array(([0,0,0],[1,0,0],[1,0,1],[0,1,0],[1,1,0],[1,1,1]))
to_pred = pd.DataFrame(g,columns=['A', 'S', 'S_a'])
prs = res1.predict(to_pred)
preds = pd.DataFrame(np.array(prs*1000), columns=['model'],
                     index=['baseline', 'just A spike', 'A spike after stim',
                            'just stim', 'just stim and A', 'stim and A spike after
preds['truth'] = b0 + np.array([0, a_b, a_b+s_ab, s_b, s_b+a_b, s_b+a_b+s_ab])
print(preds)
print('')

# evaluate the effectiveness of the stimulation on the effect of A on B
# by calculating the model's predicted increase in firing rate
```

```
eff = 1000*(prs[5]-prs[4])
print(f'The stimulation effectiveness is {np.round(eff,2)} hz')
```