**List of variables (from 1d\_clusterless\_decoder\_sungod.py) we want to specify in the wrapper script:**

Our understanding is that these variables would be specified by the wrapper and then written to one config file per decoder run. The 1d\_clusterless\_decoder\_sungod.py script would then use this config file to run a single instance of the decoder. These series of decoder runs will either loop through different decoding parameters, different days and different animals, or different position/mark shuffles.

This list basically goes through the 1d\_clusterless\_decoder\_sungod.py script line-by-line and lists the variables we may want to change. Some of these variables will be changed when we are analyzing epochs from multiple days, or running the shuffle, or testing parameters of the encoding/decoding process. I have tried to indicate which ones fall into each category.

For the position/mark shuffle we need to write a function for a circular shuffle of the position file. We will first generate a trial-shuffled position file by applying a velocity filter (to pull out movement times) from the original position file and then randomly shuffling the order of the trials. Then, we will apply 1000 circular shuffles of random time offsets (between 1/4 and 1/2 the time of the epoch) to the trial-shuffled position file. Alan, can you help us write a function to do this shuffle?

**Line-by-line list of variables:**

**line 35**: path\_main = '/home/mcoulter/spykshrk\_realtime'

**line 49**: path\_base\_rawdata = '/home/mcoulter/raw\_data/'

**line 85**: path\_base\_analysis = '/mnt/vortex/mcoulter/'

explanation: these are the paths we need to set for accessing data files.

variable category: shouldnt have to change these after setting up the data storage space at LLNL for the raw data.

**Paths not yet included in the python script**:

We also want to specify a path for the log files we will use to keep track of the data provenance for each decoder run.

These logs would include: terminal output from the wrapper script, terminal output from the 1d\_clusterless\_decoder\_sungod.py script, and the config file used by each particular run. We might want to add some error messages to these scripts to track when things go wrong. Dan has many of these in the core code, but I don't have any yet for 1d\_clusterless\_decoder\_sungod.py. Alan, can you help with this?

**lines 52-57**:

rat\_name = 'remy'

directory\_temp = path\_base\_rawdata + rat\_name + '/'

day\_dictionary = {'remy':[20], 'gus':[24]}

epoch\_dictionary = {'remy':[2], 'gus':[7]}

tetrodes\_dictionary = {'remy': [4,6,9,10,11,12,13,14,15,17,19,20,21,22,23,24,25,26,28,29,30],

'gus': list(range(6,13)) + list(range(17,22)) + list(range(24,28)) + [30]}

explanation: these are the data input variables for each epoch. we want to specify all of these parameters for each different run. There are 4 possible rat names: remy, gux, bernard, and fievel

variable category: these are variables will we want to loop through when we are running the decoder across days and across animals. Each config file will have one specific set of these parameters, such as Remy, day 20, epoch 2, tetrode 4. Or Remy, day 20, epoch 2, all tetrodes.

**line 101**: linflat\_ripindex\_encode\_velthresh = linflat\_ripindex.query('linvel\_flat < 4')

explanation: here we want to specify the velocity filter (4) for ripples

variable category: we will change this when adjusting the decoder, but won't change it for different animals or shuffles.

**lines 114,115**:

encode\_subset\_start = 0

encode\_subset\_end = 5000

explanation: specifies the time subset that will be used to build the encoder

variable category: we generally want to analyze full epochs, so this will almost always be [0:-1], but we may want to try smaller subsets when we are testing parameters of the decoder.

**line 139**: linflat\_spkindex.query('linvel\_flat > 2')

explanation: here we want to specify the velocity filter (2) for the encoder

variable category: we will change this when adjusting the decoder, but won't change it for different animals or shuffles.

**lines 148,149**:

decode\_subset\_start = 0

decode\_subset\_start = 5000

explanation: specifies the time subset that will be decoded. this may be continuous or discontinuous.

variable category: this will likely change for each epoch, because we either want to decode the whole epoch [0:-1] or just the times of ripples - the ripples times are epoch specific. in the case of only ripple times, this will be a discontinuous list of times. alan, can you set up the wrapper so that it can accept either a continuous or discontinuous array?

**line 128**: spk\_subset\_sparse = trodes2SS.threshold\_marks(spk\_subset, maxthresh=2000,minthresh=100)

**line 16**2: spk\_subset\_sparse\_decode = trodes2SS.threshold\_marks(spk\_subset\_decode, maxthresh=2000,minthresh=100)

explanation: these specify the thresholds for min and max spike amplitude (maxthresh, minthresh) and will always be changed together to the same values.

variable category: we will change this when adjusting the decoder, but won't change it for different animals or shuffles.

**lines 383-401**:

max\_pos = int(round(linear\_distance\_arm\_shift.max()/5)+5)

encode\_settings = AttrDict({'sampling\_rate': 3e4,

'pos\_bins': np.arange(0,max\_pos,1),

'pos\_bin\_edges': np.arange(0,max\_pos + .1,1),

'pos\_bin\_delta': 1,

'pos\_kernel': sp.stats.norm.pdf(np.arange(0,max\_pos,1), max\_pos/2, 1),

'pos\_kernel\_std': 1,

'mark\_kernel\_std': int(20),

'pos\_num\_bins': max\_pos,

'pos\_col\_names': [pos\_col\_format(ii, max\_pos) for ii in range(max\_pos)],

'arm\_coordinates': arm\_coordinates\_WEWANT})

#cell 9

#define decode settings

decode\_settings = AttrDict({'trans\_smooth\_std': 2,

'trans\_uniform\_gain': 0.0001,

'time\_bin\_size':60})

explanation: these variables define all the parameters for the encoder and decoder

variable category: we will change these when adjusting the decoder, but won't change them for different animals or shuffles.

**line 412**: dask\_worker\_memory=1e9

dask\_chunksize = None

explanation: either one of these specifies the dask settings; one of the two must always be set to "None". we dont know if changing dask\_worker\_memory will help decrease the run time for the decoder. this might be something to look into. alan, do you have a sense of whether this is a parameter to try changing?

variable category: we will change this when adjusting the decoder, but won't change it for different animals or shuffles.

**\*\*\* The next two variables are ones we need to replace. \*\*\***

**line 447**:

observ\_obj.\_to\_hdf\_store('/data2/mcoulter/remy\_20\_4\_observ\_obj\_0\_2000.h5','/analysis',

'decode/clusterless/offline/observ\_obj', 'observ\_obj')

explanation: specify the file when saving the observations dataframe

**line 455**:

observ\_obj = Posteriors.\_from\_hdf\_store('/data2/mcoulter/remy\_20\_4\_observ\_obj\_0\_20000.h5','/analysis',

'decode/clusterless/offline/observ\_obj', 'observ\_obj')

explanation: specify the file when loading a previously saved observations dataframe

**NOTES**:

Based on our discussion yesterday, we actually want to save "results" not "observ\_obj" (results is generated in line 416). Results is a list of dataframes with one dataframe per tetrode and these dataframes are then combined to make the observ\_obj dataframe (in lines 421-443). So, when we run the decoder on 1 tetrode, results will have 1 dataframe and we want to combine that with the results dataframes from the other 20 tetrodes that were each run separately.

Alan, Do you have good functions for saving pandas dataframes to hdf and then reloading them? Or we can use the functions that dan wrote called .\_to\_hdf\_store and .\_from\_hdf\_store. These functions are in the script data\_containers.py in the folder /spykshrk\_realtime/spykshrk/franklab. Can you help us choose good functions for this?

Variable category: we will need to specify a different file for each run of the decoder.

**lines 462-479**:

time\_bin\_size = 60

decode\_settings = AttrDict({'trans\_smooth\_std': 2,

'trans\_uniform\_gain': 0.0001,

'time\_bin\_size':60})

encode\_settings = AttrDict({'sampling\_rate': 3e4,

'pos\_bins': np.arange(0,max\_pos,1),

'pos\_bin\_edges': np.arange(0,max\_pos + .1,1),

'pos\_bin\_delta': 1,

'pos\_kernel': sp.stats.norm.pdf(np.arange(0,max\_pos,1), max\_pos/2, 1),

'pos\_kernel\_std': 1,

'mark\_kernel\_std': int(20),

'pos\_num\_bins': max\_pos,

'pos\_col\_names': [pos\_col\_format(ii, max\_pos) for ii in range(max\_pos)],

'arm\_coordinates': arm\_coordinates\_WEWANT,

'spk\_amp': 60,

'vel': 0})

explanation: we need to define/re-define these parameters to run the decoder function

variable category: we will change this when adjusting the decoder, but won't change it for different animals or shuffles.

**line 482**: trans\_mat=encoder.trans\_mat['flat\_powered']

explanation: specifies which transition matrix for the decoder to use

variable category: we will change this when adjusting the decoder, but won't change it for different animals or shuffles.

**line 492**:

posteriors.\_to\_hdf\_store('/mnt/vortex/mcoulter/posteriors/remy\_20\_2\_linearized\_0\_5000.h5','/analysis',

'decode/clusterless/offline/posterior', 'learned\_trans\_mat')

explanation: this is the final output! this variable specifies the file when saving the posteriors dataframe

Variable category: we will need to specify a different file for each run of the decoder.