Computational neurodynamics

Exercise Sheet 4 (**Assessed**)
Dynamical complexity

A good experiment can be replicated. Your task is to replicate one of the experiments described in Topic 9 (Dynamical Complexity). The experiment involves the generation of modular small-world networks of spiking neurons, and an assessment of their dynamical complexity.

Question 1.

Write Matlab code that will generate small-world modular networks of Izhikevich neurons. The basic method is the rewiring procedure described in Topic 8 (Modular Networks). But for details of the network construction, follow the description of the experiment in Topic 9 (Dynamical Complexity). Note that the networks need a separate inhibitory population. Use the neuron parameters from the earlier lecture notes for excitatory and inhibitory Izhikevich neurons. You can use `IzNeuronUpdate.m` from Murray Shanahan's Teaching page.

Generate a networks for each of the following values of p: $p = 0$, $p = 0.1$, $p = 0.2$, $p = 0.3$, $p = 0.4$, and $p = 0.5$. For each of these networks:

a) Generate a plot of the matrix connectivity

b) Generate a raster plot of the neuron firing in a 1000ms run.

c) Generate a plot of the mean firing rate in each of the eight modules for a 1000ms run. Downsample the firing rates to obtain the mean by computing the average number of firings in 50ms windows shifted every 20ms. (This should yield 50 data points for each module.)

Your results should be consistent with those in the Topic 9 slides.

Question 2.

Following the description in the slides, write Matab code that generates a network as in Question 1, for a random value of p between 0.1 and 0.5, then runs it for 60 seconds while recording the (downsampled) mean firing rates in each of the eight modules. Discard the first second's worth of data. (This should yield eight time series of 2950 data points each.)

Now write Matlab code to calculate the *neural complexity* for such a set of time series. Note that, for neural complexity to be a valid measure, it has to be applied to time series that are *covariance stationary*. To render your time series covariance stationary, apply differencing to them *twice* before calculating their neural complexity. You can use the function `aks_diff.m` (from Anil Seth), which is also available via Murray Shanahan's Teaching page.

Perform at least 20 trials as described above, and plot the results. This should resemble the plot in the Topic 9 slides.

What to Submit

You should submit all your Matlab code, plus your plots as Matlab fig files. Also submit a README.txt file which specifies the name of the top-level Matlab function call for each question, as well as the file names of the plots associated with each question.

Working in Groups

You are encouraged to work in groups of up to three students. But you may work in pairs or alone if you prefer.

Marking Scheme

70% of the marks will be allocated to Question 1, and 30% to Question 2. Note that this is not a direct reflection of the relative difficulty of the two questions. Rather, it allows students to gain extra marks on this coursework for extra effort (on Question 2), while allowing those who want to balance their workload with other courses to put in less effort (on Question 2) without significantly compromising their grade.

Frequently Asked Questions

QUESTION: Isn't the description of how you build a modular small-world network in the Topic 8 slides slightly different from the description in the relevant Topic 9 slides?

ANSWER: Yes! According to the notes from Topic 8, to build a modular small-world network with n nodes, m edges, and C communities, you first construct C disconnected communities with n/C nodes each, where each community has m/C random symmetric edges. According to the exercise specification, you should try to reproduce the results in Topic 9, where "each excitatory neuron has five outgoing connections to other excitatory neurons" and "connections are not symmetrical, so we have a directed network". For the exercise, you should follow the second description, from Topic 9.

QUESTION: What scaling factors / weights / conduction delays should we use?

ANSWER: These are all specified on a slide in Topic 9 (Dynamical Complexity).

QUESTION: What are layers for? How should we use them?

ANSWER: The layers (used in IzNeuronUpdate.m) are only there as a software engineering aid to help organise large networks. You could put all the neurons in a single layer. But in the case of the exercise, a natural organisation is to put all the excitatory neurons in one layer and all the inhibitory neurons in another. The modules (which comprise excitatory neurons) are then all within a single layer.

QUESTION: Neurons in the code are organised as N by M matrices, although M is always set to 1 in the sample code. Should we use the second dimension?

ANSWER: No. Ignore this feature, and just use 1D arrays of neurons by always setting M to 1, as in the sample code.