Assignment 5: Introduction to Nengo

This assignment should be done in pairs. Do not forget to put both your names on the submission.
Using Large Language Models (e.g. ChatGPT) in this assignment is not allowed.

Goal

The goal of this assignment is to learn and understand how values are represented and communicated in Nengo. In the first part, you will use neural ensembles to represent, transfer, and manipulate values in the Nengo interface. In the second part, you will recreate parts of this model in default Python, with the goal to see the underlying computations.

Running Nengo

Nengo is a Python package that uses a browser-based GUI. To run Nengo on university computer, boot into the Linux distribution (LWP), open a terminal window, and type `nengo`. This will open a browser window of your default browser showing the Nengo GUI.

To run Nengo on your own computer you can follow these instructions: https://www.nengo.ai/getting-started/ (you might want to check out the *Detailed Installation Instructions* using Anaconda or miniconda). It runs on Mac, Windows, and Linux, and only requires Python and Numpy. **However, for me it throws an error when running it on a Python version >3.9.**

If you already have conda installed on your computer, these are the steps to create a new environment and install Nengo:

1. Create environment with Python 3.9 and pip, and activate it:

```
conda create --name nengo-env python=3.9 pip
conda activate nengo-env
```

2. Install nengo:

```
pip install nengo nengo-gui
```

3. Run nengo:

```
nengo
```

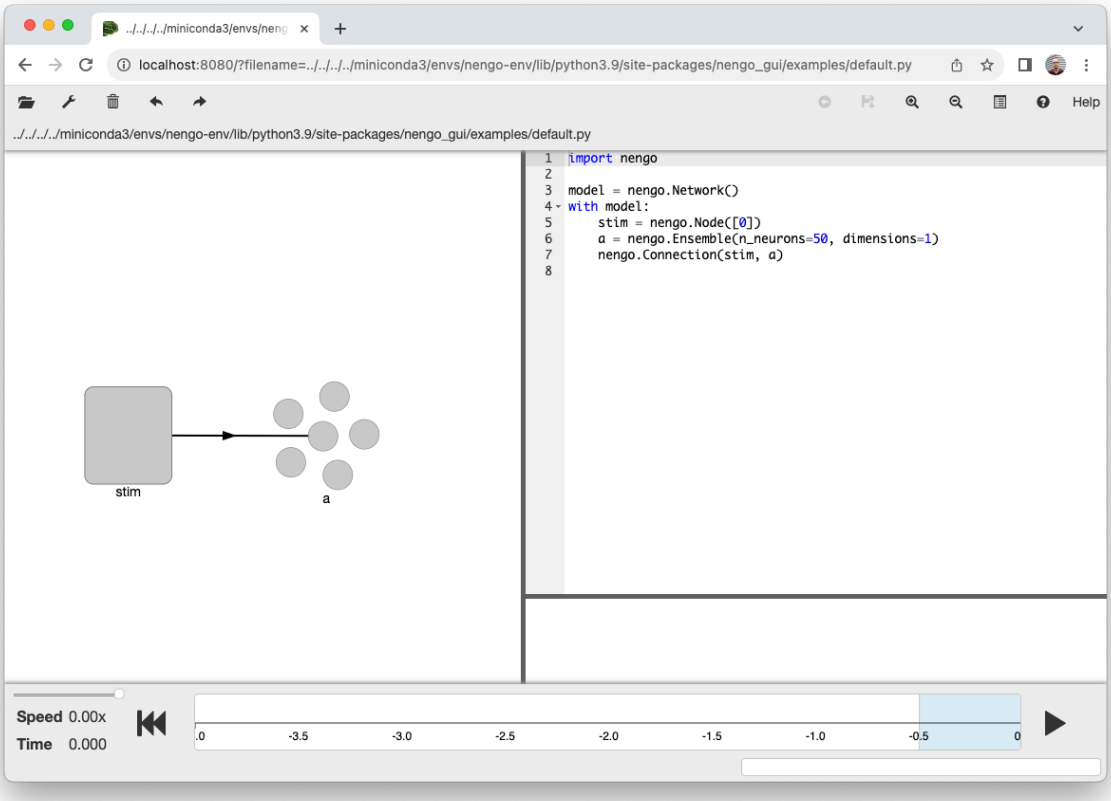
In my experience, Google Chrome works best as the browser:

```
nengo --browser chrome
```

If you have issues with numpy imports, it works with the version 1.26.2

```
pip install numpy==1.26.2
```

Once the GUI is up and running, the frame on the right shows the python code of your current model, and the frame on the left a graphical representation:



Assignment Part 1: Values and Communication in the Nengo GUI

In the Nengo GUI, if you click on the "open" icon in the top-left, select *built-in examples* and then *tutorial* you will find a list of tutorial models that cover a wide variety of Nengo functionality.

For part 1, start by doing tutorials 00-08 and 11. This provides both a reminder of some of the things discussed in class and example code that provides the required programming syntax. Do read all comments, and play around with the code to see what happens when you make changes to, for example, the number of neurons.

Now build your own model, consisting of one `stimulus` node and three ensembles, `a`, `b`, and `c`:

- The stimulus `Node` has a value of `[0]`
- Each `Ensemble` consists of 100 neurons and represents one value (`dimensions=1`).

Connect the stimulus and ensembles as follows:

- `stimulus` to `a` is a simple connection ***without*** transformations or functions.

- `a` to `b` calculates the square of `a` centered around 0: `a*a - .5`.
- `b` to `c` is a connection that only gives half input to `c`, by specifying `transform=.5`.
- make `c` into a simple memory by specifying a recurrent connection from `c` to `c`. As in tutorial 11, set the synaptic constant of this connection to 100 ms.

Play around with the stimulus, check if everything works as you expect. When it does, add an Ensemble `d` that represents two values at once by using two dimensions. Connect the stimulus to the first dimension of `d`, and ensemble `c` to the second dimension of `d`.

Now, use a sine as the input for the stimulus node, by defining a sine function:

```
def sin_func(t):
    return np.sin(2*t)
```

and specifying the node as `nengo.Node(sin_func)`.

What to hand in for Part 1:

- a python file with your model, with comments that explain the code inside the "`with model:`" part. You can add comments to python using `#`-marks.

- a PDF with:

    a. A screenshot in which you show values (right click, Value) for ensembles `a`, `b`, and `c`, and a two-dimensional value for ensemble `d` (XY-value) while running your model, demonstrating the effects of the various connections that you made.
    b. A screenshot of the tuning curves of ensemble `a` (right click on `a`, Details..., Plots). Explain what those tuning curves indicate.
    c. Set the stimulus to 0 (right-click on the stimulus, 'Set value...' – it will ignore the sine function) . Explain the pattern that you see in the value displays of `a`, `b`, and `c`.

Assignment Part 2: Values and Communication in Python

In this part of the assignment, we are going to look at how Nengo builds ensembles and connections under the hood. For this, you will work directly in Python. While you can just copy-paste commands into a Python shell, I would advise you to work in an IDE like Spyder (https://www.spyder-ide.org; `pip install spyder`), PyCharm (https://www.jetbrains.com/pycharm/), or VS Code (https://code.visualstudio.com). The only extra library you need is matplotlib (`pip install matplotlib`).

First, download nengo_in_python_AB.py. This files implements the ensembles A and B discussed above, including the connection between them. Read it through carefully, try to understand what the functions do and how the final results come about. Note

that this code is clearly less efficient than the actual Nengo implementation, even though the tuning curves are much less smooth. Nengo includes quite a number of smart tricks to make it possible to run large models.

Second, add an ensemble C (consisting of 50 neurons) and the connection between B and C, multiplying the value in B by 0.5 (equivalent to the transform in Part 1). You do **not** need to add the recurrent connection for C.

Third, add an output graph in which you show the ideal signal in C and the represented signal, just like it was done for ensembles A and B.

What to hand in for Part 2:

- the resulting Python file, which simulates ensembles A, B, and C.

- a PDF with:
    a. the graph with the ideal and actual signals in ensemble C.
    b. an explanation of the functions `compute_tuning_curves`, `compute_response`, and `run_neurons` at a conceptual level. That is, explain in your own words what the inputs and outputs of those functions are, and how they are computed. You do not have to explain gain and bias, these are properties of the neurons, defining their exact responses to input.

Bonus (2 points)

You can earn 2 extra points by also adding ensemble D of Part 1 to the Python code of Part 2. Note that this ensemble – that is, the neurons – is 2-dimensional, and receives input both from ensemble C and the original sine input. This is quite difficult.

What to hand in?

One zip file, containing everything described under **What to hand in for Part 1** and **What to hand in for Part 2**.

Where to hand in?

Use this assignment item to submit your work. The deadline is strict. We do not accept submissions by email.

Deadline

Monday December 16, 09:00h.