

### ***Milestone 4.3.3 -- Adaptive Motor Control in Spaun***

***Xuan Choo and Chris Eliasmith***

***Milestone 4.3.3:*** Integrate adaptive motor controller into a next generation Spaun.

#### **1.0 Introduction**

In deliverable 4.3.2 we described the introduction of an improved visual system into the large-scale Spaun model (Eliasmith, et al., 2012). As noted, during the course of this project, we have developed several spiking network algorithms that significantly improve upon those used in the original Spaun. In this deliverable, we focus on the integration of our improved motor system into Spaun (see deliverables 4.1.1-4.1.4).

Deliverable 4.1.1 describes the REACH model in detail, which forms Spaun's new motor system, which controls a single arm as in the original Spaun. As shown in that deliverable, this new motor system improves on the original system in several respects, and demonstrates good quantitative matches to several aspects of human reaching. Most importantly, the new motor system is fully implemented in spiking neurons and includes the ability to adapt to unknown forces on-the-fly. Neither of these are true of the original Spaun motor system.

Deliverables 4.1.2-4.1.3 focus on performing a wide variety of motor control experiments and comparing the results to state-of-the-art deep learning methods and state-of-the-art (non-neural) control on the same tasks (4.1.4 implements the controller on a physical robot). As described in detail in deliverable 4.1.3, the REACH model is as good or better than other approaches across the tasks (see, e.g., Figures 1A, 1B, 5 and 7). Furthermore, the REACH model is more computationally efficient and scales better than these other approaches, suggesting it can effectively exploit the strengths of neuromorphic hardware. One of the tasks considered in these deliverables is the "force field" task, which is the focus of our demonstration of integrating this model into Spaun, as it requires on-the-fly adaptation which could not be captured by the previous Spaun model.

Consequently, the focus of this report is on demonstrating the effective adaptation of the fully spiking motor model integrated into Spaun. Technical details of the REACH model can be found in deliverable 4.1.3 and (DeWolf, 2014). Here we describe the main results of integrating the new motor system and detail methods for reproducing these results in the Nengo neural simulator.

#### **2.0 Results**

To demonstrate the adaptive characteristics of the new motor system in Spaun, the force field task has been adapted to number writing. The original task is described in deliverable 4.1.2 and 4.1.3 reproduces a series of experiments performed on human subjects (Shadmehr, 1994).

This task is similar to a standard center-out reaching task, in which the subject's hand begins at a center position, and then they reach to one of several (typically 8) endpoints. In the case of the

force field task, a perturbing force field is introduced. This is typically accomplished by having subjects hold on to an object (e.g. joystick) under the control of the experimenter, while making the movements. The object is then subjected to forces that can be a function of position or other reach parameters. In the experiments we are reproducing, the force field is a function of the end effector velocity, making for a more challenging reaching task than a static (i.e., position dependent) force field. This means that the dynamics of the system change significantly over the trial.

In human trials on this task, adaptation was considered complete when the correlation between the non-force field reach paths and force field reach paths was over 0.9. As described in deliverable 4.1.3 this performance was matched by the REACH model along several dimensions (e.g., adaptation accuracy, reach paths, neural activity, etc.). As shown in Figure 1, comparison controllers did not perform as well.



Figure 1: Force field reaching performance of several controllers compared. The REACH model performance (post adaptation) is shown on the far right. The deep learning (hf: hessian free) model (far left) generalizes poorly to other situations and training time proved to be too large for comparison of this controller on each of the tasks, we only show results for basic reaching. ilqg: iterative linear quadratic gaussian. lqr: linear quadratic regulator. osc: operational space control.

To demonstrate integration of the REACH model into Spaun, we have implemented the same kind of force field task, but for number drawing rather than straight line reaching. Since number drawing requires more sophisticated paths, this is a more difficult task than simple reaching. The As the adaptive controller requires multiple trials to improves the fidelity of the written digits, Spaun is tasked with repeated instances of the list recall task (the task with the shortest amount of input stimuli required per number of digits in Spaun's response).

Figure 2 shows Spaun's responses to the task of remembering and reproducing the list [4, 2, 7, 5] (numbers chosen at random) in the presence of the end-effector velocity modulated force field used in Figure 1. For the responses illustrated in Figure 2, the adaptive controller has been removed to demonstrate the baseline response Spaun's default motor controller (OSC) produces in the presence of the force field.

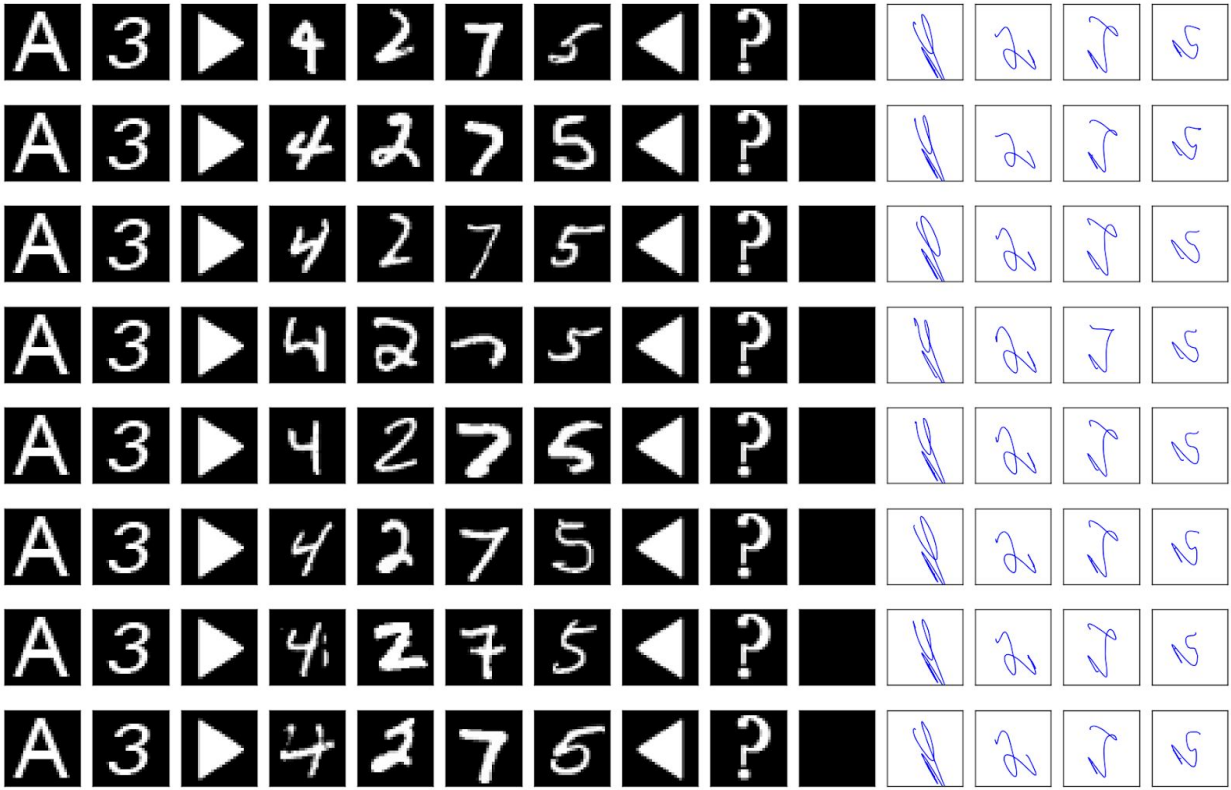


Figure 2: Baseline digit responses of Spaun's default motor controller for 8 consecutive trials of the memory task in the presence of the velocity-modulated force field. The input stimuli is shown as white characters in the black squares, while Spaun's motor response is illustrated as blue paths in the white squares.

From Figure 2, it can be seen that without the adaptive controller, Spaun's written responses are consistent, barely legible and do not improve with the number of trials. Figure 3 shows the same task, but with the adaptive controller included in Spaun's motor system. From the figure, it can be seen that over the course of the trials, the adaptive controller learns the effect of the force field and gradually improves the accuracy of Spaun's written responses.

It should be noted that the 7th trial of the memory task shown in Figure 3, Spaun mis-classifies the input digit 4 as the number 6, and thus responds with the list [6, 2, 7, 5] rather than the correct answer of [4, 2, 7, 5]. However, this mistake illustrates that the adaptive motor controller has not learned just to accurately reproduce the specific digits 4, 2, 7, and 5, but rather, it has learned to generalize the effect of the velocity modulated force field and is thus able to appropriately compensate when reproducing any written response.

As a point of reference, Figure 4 illustrates Spaun's written responses using the default OSC motor controller without the adaptive motor controller nor the force field applied to the system.

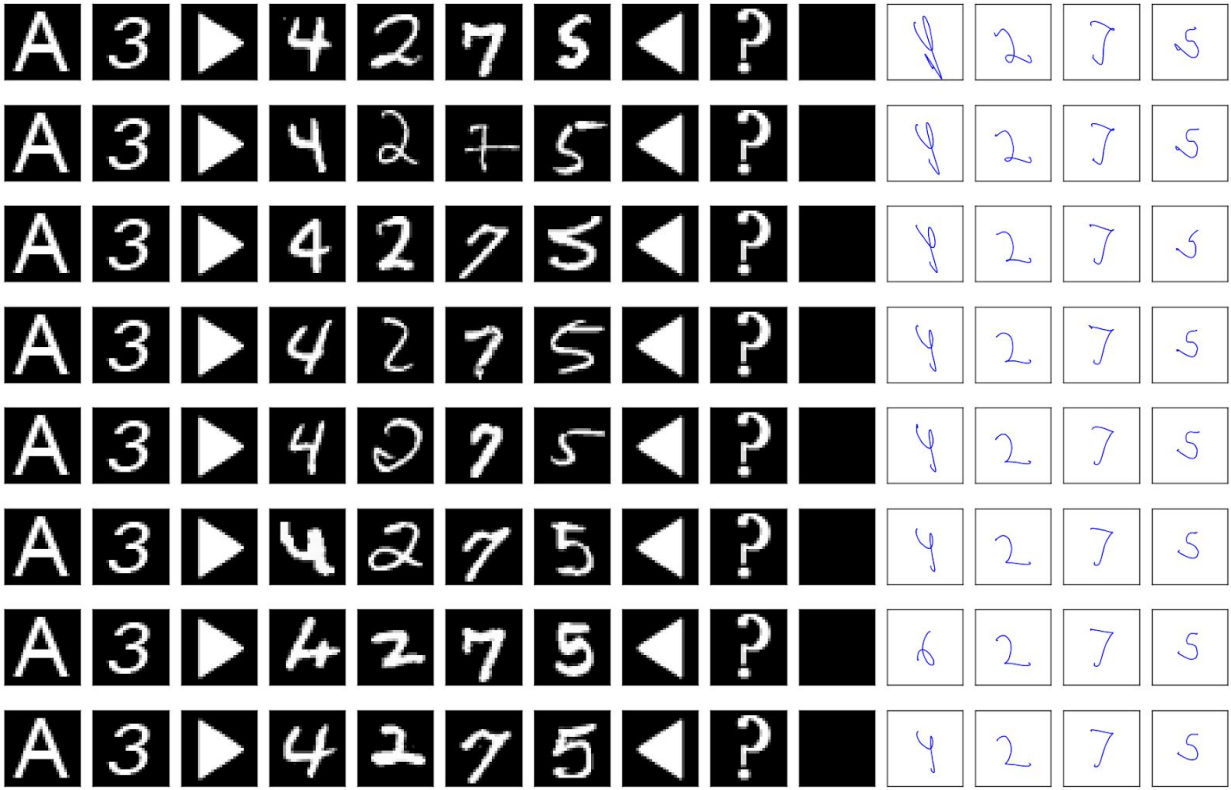


Figure 3: Digit responses of Spaun's new adaptive motor controller for 8 consecutive trials of the memory task in the presence of the velocity-modulated force field. As with Figure 2, the input stimuli is shown as white characters in the black squares, while Spaun's motor response is illustrated as blue paths in the white squares. Note that for the 7th trial (row 7), Spaun wrongly identified the input stimulus (the digit 4) as a 6, and as such reproduces the list [6, 2, 7, 5] instead of the correct response of [4, 2, 7, 5].

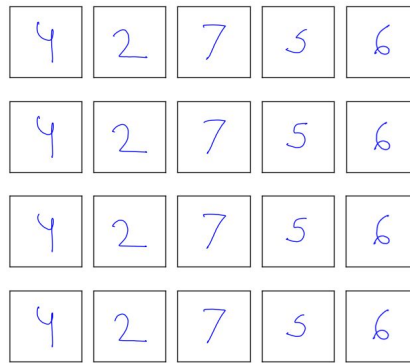


Figure 4: Examples of Spaun's written responses for the digits 4, 2, 7, 5, and 6 using Spaun's default motor controller without adaptation, nor the influence of the velocity modulated force field.

An additional significant improvement over the original Spaun model is that with this new motor system, the entire system is a spiking neural network. Figure 5 shows plots of the spiking neural activity of the cerebellar (CB), M1 (M1) and adaptive controller (Adapt Ens). Using the Neural Engineering Framework, it is possible to decode the spiking activity (see the Adapt Ens value plot) of the adaptive controller to get a sense of the function it is representing. For the memory tasks illustrated in Figure 5, a constant force field (constant force of 0, 100 and 200 units of force) has been applied to each joint in Spaun's arm. As illustrated in Figure 5, after 2 consecutive trials of the memory task, the adaptive controller has learned to compensate for this force field -- the decoded value of 'Adapt Ens' stabilizes at 0, -100, and -200 units of force, effectively cancelling out the effect of the constant force field.

Figure 6 shows the same data recorded for Figure 5, graphed for the first 4 seconds of the simulation. This was done to make individual spikes more distinct in the final figure.

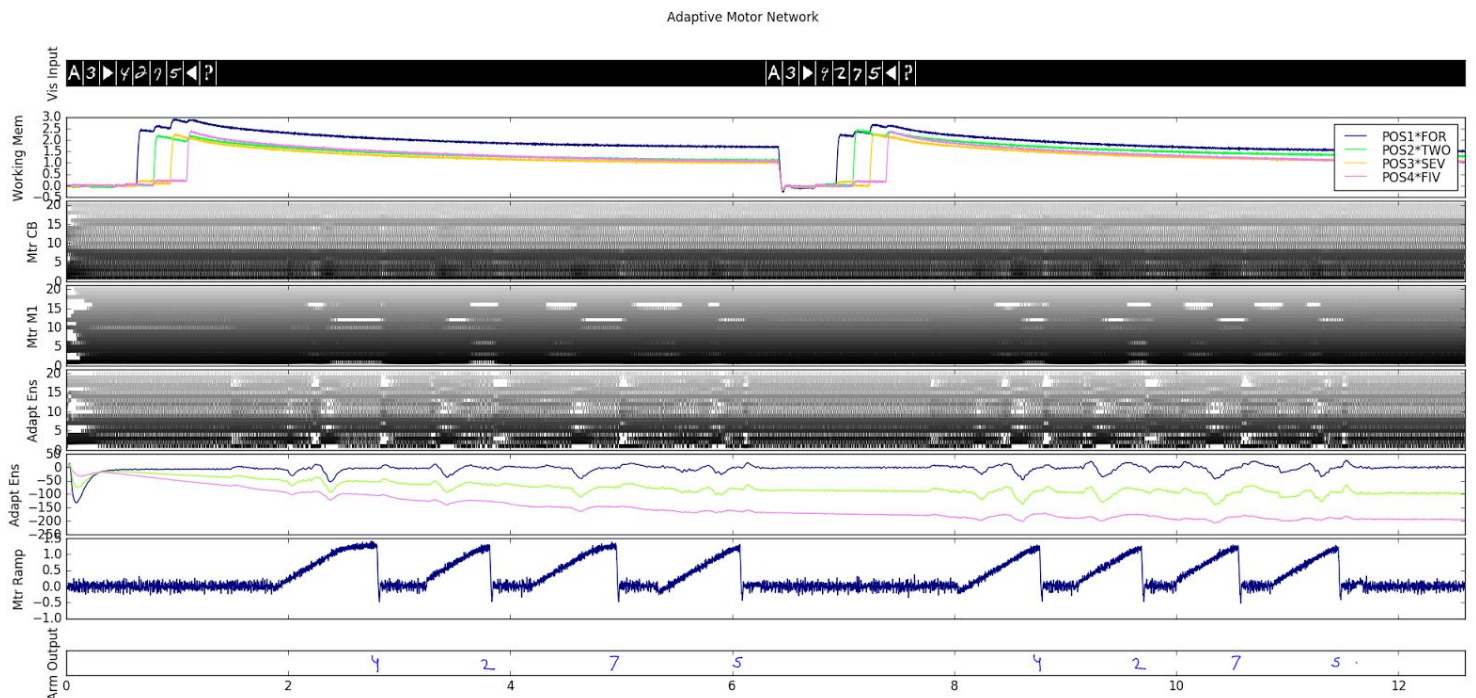


Figure 5: Spiking activity of motor control system with the adaptive controller performing two consecutive trials of the list memory task. Shown are the visual stimuli that Spaun was presented (Vis Input), the semantic pointer representation of the values stored in Spaun's working memory (Working Mem), the spikes recorded from the cerebellar (CB), M1 (M1) and adaptive controller (Adapt Ens) populations within the motor control system, the 'decoded' output of the adaptive controller (Adapt Ens) population, the ramp signal driving the motor system output (Mtr Ramp), and Spaun's written responses (Arm Output).



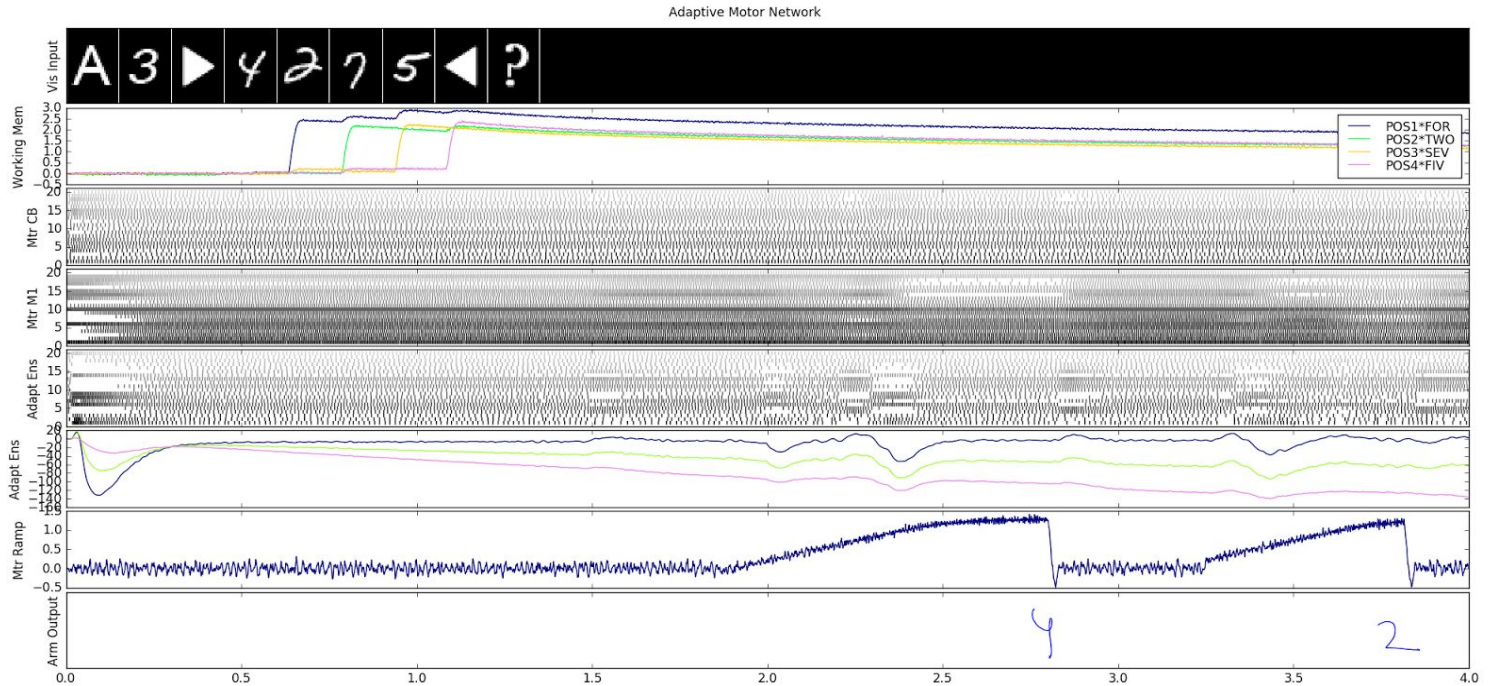


Figure 6: Spiking activity of motor control system with the adaptive controller performing two consecutive trials of the list memory task. The data used in this plot is identical to Figure 5, with the plot showing only the first 4 seconds of the task.

### 3.0 Usage

The new version of Spaun is built in Nengo 2.0 (Bekolay et al., 2014). To install the Nengo GUI, Python must be available. Then at a command prompt the following command will install the simulator (for detailed instructions, see [https://github.com/nengo/nengo\\_gui](https://github.com/nengo/nengo_gui)):

```
pip install nengo_gui
```

It is also recommended to install `nengo_ocl` as well to decrease the time needed to run the simulations (for detailed instructions, see [https://github.com/nengo/nengo\\_ocl](https://github.com/nengo/nengo_ocl)):

```
pip install nengo_ocl
```

The Spaun model itself can be obtained from the experimental's branch in Spaun github repository. It is also recommended to install matplotlib to see the plots that the various scripts generate:

```
pip install matplotlib
cd ~/git
git clone https://github.com/xchoo/spaun2.0
cd spaun2.0
git checkout experimental
```

To run Spaun with the options for using / not using the motor adaptation for the different types of force field tasks, use the `--config` option of the `run_spaun.py` script. The parameters that the `--config` option supports (for the adaptive force field task) are `mtr_dyn_adaptation` and `mtr_forcefield`. The following is a list of possible values that can be used with these parameters:

- To use the adaptive motor controller, use: `"mtr_dyn_adaptation=True"`
- To disable the adaptive motor controller, use: `"mtr_dyn_adaptation=False"`
- To run Spaun's tasks with the joint-velocity-modulated force field, use (note: both sets of quotations are needed for this to run properly):  
`"mtr_forcefield='QVelForcefield'"`
- To run Spaun's tasks with the end-effector-velocity-modulated force field, use:  
`"mtr_forcefield='XYVelForcefield'"`
- To run Spaun's tasks with the constant-joint-force force field, use:  
`"mtr_forcefield='ConstantForcefield'"`
- To run Spaun's tasks with the no force field, either omit this parameter, or use:  
`"mtr_forcefield='NoForcefield'"`

Additionally, to automatically display the plots shown in Figure 2 - 4, use the `--showiofig` argument option, and to display the graphs shown in Figure 5 & 6, use the `--showgrph` argument option.

For example, to run the Spaun simulation to generate the graphs for Figure 2 (velocity modulated force field with no adaptation), run:

```
cd ~/git/spaun2.0
python run_spaun.py -s '{A3[#4#2#7#5]?XXXX:8}' --showiofig
--config "mtr_dyn_adaptation=False"
"mtr_forcefield='QVelForcefield'"
```

Likewise, to run the Spaun simulation to generate the graphs for Figure 3 (velocity modulated force field with adaptation), run:

```
python run_spaun.py -s '{A3[#4#2#7#5]?XXXX:8}' --showiofig
--config "mtr_dyn_adaptation=True"
"mtr_forcefield='QVelForcefield'"
```

To generate the spike plots shown in Figure 4 (no force field with no adaptation), use the `--showgrph` argument (instead of `--showiofig`), omit the `mtr_dyn_adaptation` and `mtr_forcefield` parameters and include the `probe_graph_config` parameter with the value `'ProbeCfgDarpaMotor'` and run:

```
python run_spaun.py -s '{A3[#4#2#7#5]?XXXX:8}' --showgrph
--config "probe_graph_config='ProbeCfgDarpaMotor'"
```

To generate the spike plots shown in Figure 5 (constant force force field with adaptation), run:

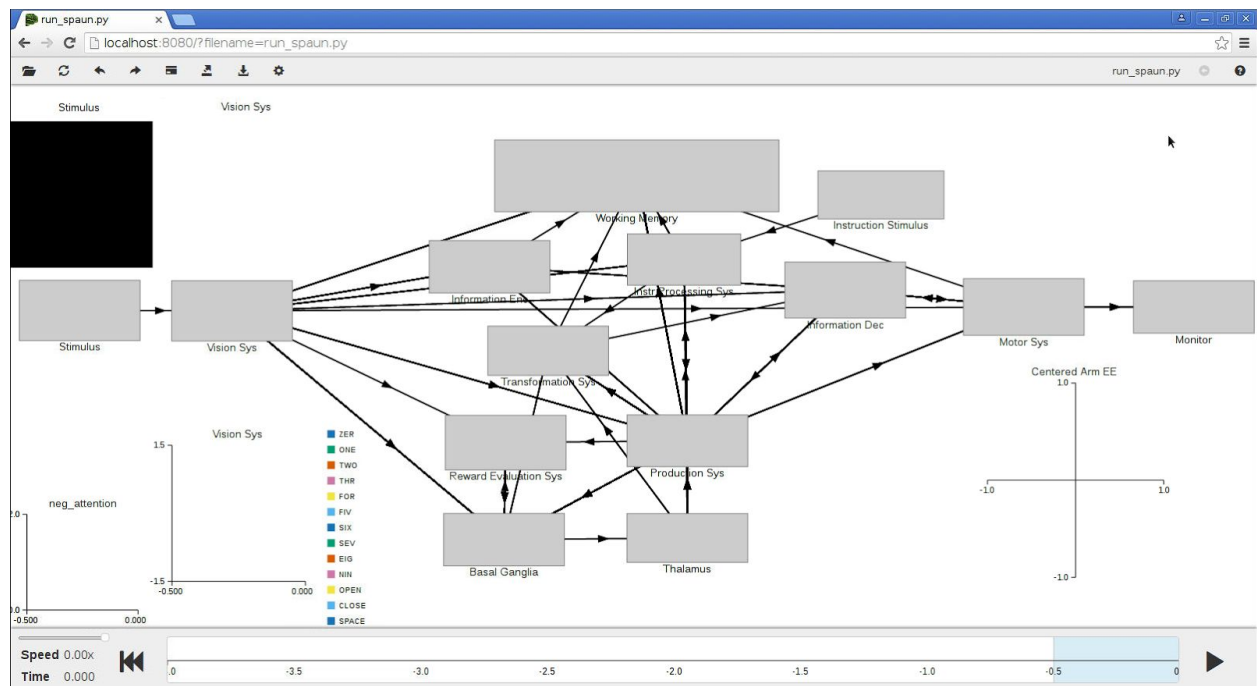
```
python run_spaun.py -s '{A3[#4#2#7#5]?XXXX:2}' --showgrph
--config "probe_graph_config='ProbeCfgDarpaMotor'"
"mtr_dyn_adaptation=True"
"mtr_forcefield='ConstantForcefield'"
```

Lastly, to generate the spike plots shown in Figure 6 (same as Figure 5, but first 4 seconds), use the `-t` argument option, and run:

```
python run_spaun.py -s '{A3[#4#2#7#5]?XXXX:2}' --showgrph
-t 4 --config "probe_graph_config='ProbeCfgDarpaMotor'"
"mtr_dyn_adaptation=True"
"mtr_forcefield=ConstantForcefield"
```

Spaun can also be visualized and run in `nengo_gui` by adding the `--nengo_gui` option to the `run_spaun.py` script command line.

```
python run_spaun.py --nengo_gui
```



In addition to the command line argument options above, the time it takes to simulate Spaun can be decreased by running Spaun with `nengo_ocl`. To do so, have `nengo_ocl` installed (see above for details), and then provide the `--ocl` argument to the `run_spaun.py` script. OCL platform and device settings can be specified using the `--ocl_plaform` and



--ocl\_device argument options respectively (running the script without these options set will give you an option to manually specify these options):

```
python run_sp aun.py -s '{A3[#4#2#7#5]?XXXX:8}' --ocl
--ocl_plaform 0 --ocl_device 0 --showiofig --showgrph
--config "probe_graph_config='ProbeCfgDarpaMotor'"
"mtr_dyn_adaptation=True" "mtr_forcefield='QVelForcefield'"
```

## 4.0 Discussion

We have demonstrated the integration of the REACH adaptive motor control system for arm movements into the Spaun model. We demonstrated its performance on the number drawing classification task while a force field was applied to the arm. This task is using the new visual system integrated in deliverable 4.3.2 for classification and the new adaptive motor system for arm control.

## 5.0 References

- DeWolf, Travis (2014). A neural model of the motor control system. PhD thesis, University of Waterloo. URL: <https://uwspace.uwaterloo.ca/handle/10012/9089>.
- Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science*, 338: 1202–1205. doi:10.1126/science.1225266
- Shadmehr, R., & Mussa-Ivaldi, F. A., (1994). Adaptive representation of dynamics during learning of a motor task. *The Journal of Neuroscience*, 14 (5 Pt 2): 3208–3224.