Spiking Neurons and Central Executive Control: The Origin of the 50-millisecond Cognitive Cycle

Terrence C. Stewart (tcstewar@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

Abstract

A common feature of many cognitive architectures is a central executive control with a 50-millisecond cycle time. This system determines which action to perform next, based on the current context. We present the first model of this system using spiking neurons. Given the constraints of well-established neural time constants, a cycle time of 46.6 milliseconds emerges from our model. This assumes that the neurotransmitter used is GABA (with GABA-A receptors), the primary neurotransmitter for the basal ganglia, where this cognitive module is generally believed to be located.

Keywords: cognitive cycle time; central executive; LIF neurons; neural production system; neural engineering framework; cognitive architectures

Introduction

ACT-R, Soar, GOMS, EPIC, and a variety of other approaches to modelling human cognition all contain a common assumption about the central control of cognitive operations. This is usually regarded as a production system where IF-THEN rules are applied sequentially. This imposes a serial bottleneck where low-level decisions as to which cognitive action should be performed next are made, requiring approximately 50 milliseconds per decision (Anderson et al., 1995).

While this 50 millisecond cognitive cycle time leads to models that match empirical data, the neurological basis for this time constraint has not been previously established. This paper develops a model of low-level rule application using realistic spiking neurons. The 50 millisecond cycle time is then shown to be the result of well-established neuron membrane and neurotransmitter properties. The result is not only a realistic, neurally plausible model of a core component for cognition, but also an explanation for why this characteristic time appears across architectures.

The model presented here is not meant to be complete. In particular, we do not provide a model of the developmental process which leads to the decision making system. We also do not currently include any learning capabilities, although this is part of our ongoing research. Instead, our model uses fixed mathematically derived synaptic connection weights, in contrast to most neural network models. These derived weights are meant to be the final result of a learning process, and weights derived in this manner have been shown to be realistic and highly robust to noise and neuron death (Eliasmith & Anderson, 2003).

Recent results have suggested that the brain area that corresponds to the system we are modelling is the basal ganglia (e.g. Anderson et al., 2004). This provides us with constraints as to the neural properties and neurotransmitters involved. However, since we are not yet modelling all aspects of this system and its interactions with other brain areas, we do not present our work as a complete model of the basal ganglia.

We start by describing the basic components of our model: the standard leaky integrate-and-fire (LIF) neuron and a model of post-synaptic current caused by a neural spike. From these, we construct a simplistic minimal model of neural decision making. We then add a competition system so that only one option at a time is represented. Finally, we build a working memory system so that context can be stored over time.

Neural Model

The standard basic model of spiking neurons is the leaky integrate-and-fire (LIF) model. While computationally simple, it provides a good approximation to real neurons over a wide range of conditions (Koch, 1999). It uses a point neuron, as opposed to more complex compartment models where ion flows within the neuron are modelled at a sub-millisecond level. Current is constantly leaking out of the neuron as per the membrane resistance *R*, but if enough input current is gathered to cause the voltage to be above a certain threshold, then the neuron will fire. After firing, the voltage is set to 0 for a fixed refractory period (2 milliseconds) before starting to gather current again.

Given a constant current input *J* and membrane resistence *R*, the voltage level of the LIF neuron changes over time as given in Equation 1 and shown in Figure 1. The timing of this behaviour is controlled by τ_{RC}, the membrane time constant of the neuron. Larger values cause the neuron's voltage to change more slowly, making it slower to respond to changes in input current. Interestingly, many real neurons are well-characterized by LIF neurons with membrane time constants in the range of 20 milliseconds, so this value is used for all simulations reported here.

$$V(t) = JR\left(1 - e^{-t/\tau_{RC}}\right) \quad (1)$$

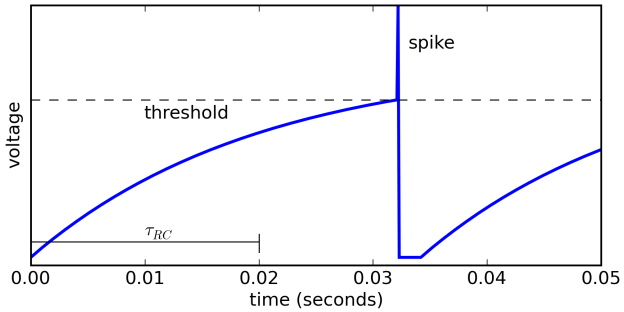

Figure 1: LIF neuron with constant input current.

When a neuron fires, it affects the input current to all of the neurons to which it is connected. This current *h(t)* can be characterized by Equation 2, where τ_s captures the effects of neurotransmitter re-uptake and dispersal. As shown in Figure 2, a small τ_s provides a fast, short-lasting effect (~10ms), while others last for hundreds of milliseconds.

$$h(t) = t e^{-t/\tau_s} \quad (2)$$

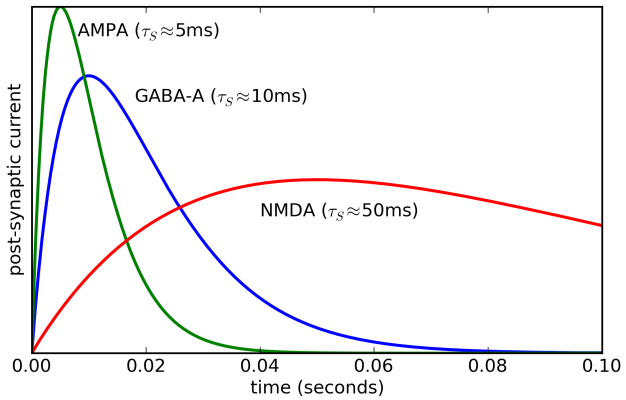

Figure 2: Post-synaptic currents for common synapses.

The τ_s values used here are approximate, based on available neurophysiology data. Gupta et al. (2000) estimate τ_s for GABA-A to be 10.41ms. AMPA is generally found to be between 1ms and 10ms and NMDA between 50ms and 150ms (e.g. Moreno-Bote & Parga, 2004).

While the neural model we are using is not a perfect replication of real neurons, we find it sufficient for our purposes. The LIF neurons allow us to explore the timing of neural processing, unlike the typical rate neurons used in most neural models. These not only do not spike, but also do not have any temporal dynamics at all, responding instantly to any change in input. Furthermore, given that the basic neural behaviour is well captured by the LIF model, switching to a more detailed model should not significantly impact the large-scale behaviour of the system (on the order of tens of milliseconds). That said, our model does not rely on the use of LIF neurons, and other more complex models could be used.

Neural Representation

Any model of a central executive control system where particular actions are chosen based on the current context must confront the issue of neural representation. The current context must be represented in such a manner as to appropriately affect the behaviour of other neurons. The approach described here to define and create such models is known as the Neural Engineering Framework (NEF; Eliasmith and Anderson, 2003).

To be as general as possible, we make the minimal assumption that representations can be distributed across a group of neurons, but leave open the question of the exact nature of this distribution. Within a neural group, each neuron has a preferred value $\tilde{\phi}$ to which it responds most strongly, and this response is reduced as the difference between this preferred value and the actual value increases.

If we assume that any value the neurons can represent can be thought of as a vector $\boldsymbol{x}$, this behaviour can be captured in terms of the input current J as shown in Equation 3. Adjusting the neuron gain α, the background input current J_{bias}, and the preferred direction vector $\tilde{\phi}$ allows us to capture a wide range of known neural representation schemes.

$$J = \alpha \tilde{\phi} \cdot \boldsymbol{x} + J_{bias} \quad (3)$$

In the simplest case, 100 neurons could represent a 100 dimensional vector $\boldsymbol{x}$ by having each $\tilde{\phi}$ be a different unit vector in each of the 100 dimensions. This would provide a completely local representation of each value in the vector. More realistically, 100 neurons could represent one or two dimensions by having $\tilde{\phi}$ values chosen randomly (i.e. uniformly distributed around the unit hypersphere in that many dimensions). This approach has been observed in numerous areas of visual and motor cortex (e.g. Georgopoulos et al., 1986). By having more neurons per dimension, the representation error can be decreased to arbitrarily low levels (error is inversely proportional to the number of neurons).

Since Equation 3 can be used as the input to a model of an LIF neuron, we can determine the sequence of spikes that would be generated for a group of neurons if a particular vector $\boldsymbol{x}$ is being represented. We can also perform the opposite operation: given a sequence of spikes we can estimate the original vector. As shown elsewhere (Eliasmith & Anderson, 2003), this can be done by deriving the decoding vectors ϕ as per Equation 4, where a_i is the average firing rate for neuron i with a given vector $\boldsymbol{x}$, and the integration is over all values of $\boldsymbol{x}$.

$$\phi = \Gamma^{-1} \Upsilon \quad \Gamma_{ij} = \int a_i a_j \, dx \quad \Upsilon_j = \int a_i x \, dx \quad (4)$$

The resulting vectors ϕ can be used to determine an estimate of the represented value using Equation 5. This is an estimate that varies over time based on the individual spikes. Importantly, it is the optimal estimate when under the constraint that the estimate must be built by linearly adding the effects of the post-synaptic currents caused by each spike. This is precisely the constraint that other neurons are under, so Equation 5 indicates the best that the original vector can be reconstructed by another neuron.

$$\hat{\boldsymbol{x}}(t) = \sum_{i,n} \delta(t - t_{i,n}) * h_i(t) \phi_i = \sum_{i,n} h(t - t_{i,n}) \phi_i \quad (5)$$

This result further provides a method for determining optimal synaptic connection weights between groups of neurons if one group is to perform a linear transformation on the value represented by the other. If one group of neurons represents $\boldsymbol{x}$ and the other group should represent $M\boldsymbol{x}$, then this can be achieved by setting the connection weights w as per Equation 6.

$$w_{ij} = \alpha_j \tilde{\phi}_j \boldsymbol{M} \phi_i \quad (6)$$

We can also use a variant of Equation 4 to determine connection weights for arbitrary nonlinear transformations of $\boldsymbol{x}$ (see Eliasmith & Anderson, 2003 for details).

The Task

As a baseline for the construction and demonstration of our model, we use a simple minimal sequential decision making task. This is meant to show that the model is capable of responding appropriately to different contexts, and is capable of modifying the context itself.

The current context is represented by a large group of neurons (at least 2000 in all models shown here), as per the representation system described in the previous section. The preferred direction vectors $\tilde{\phi}$ are chosen randomly from the unit hypersphere, and the neuron gains α and background currents J_{bias} are chosen to give a uniform distribution of maximum firing rates between 100Hz and 200Hz and an average background firing rate of 40Hz, consistent with many cortical neurons. At the beginning of a simulation, this context is fixed to represent the initial state of the model, but after this initialization period (50ms) there is no external input. That is, the model must be capable of maintaining and changing its own internal state.

We arbitrarily choose five vectors to represent five different internal states referred to as *A*, *B*, *C*, *D*, and *E*. The model's task is to implement the set of state change rules that will cause it to cycle through these five states. If the system is in state *A*, it should change to state *B*; if it is in *B*, it should change to *C*, and so on.

In terms of the cognitive architecture ACT-R, this would involve five production rules. Each production rule would match on a particular goal buffer state (*A* through *E*), and if that production fires it would modify the goal buffer to contain the next state in the sequence. In ACT-R (and in most other cognitive architectures), this process is externally fixed to require 50 milliseconds. As will be seen, in our models this timing will emerge from neural properties.

Figure 3 shows an idealized (non-neural) model of this process. The five different colours indicate the five different representational states over a period of 500 milliseconds. This is enough time for the system to repeat the cycle twice. At each moment in time, we measure the represented vector $\boldsymbol{x}$ and compare it to the arbitrarily chosen patterns *A* through *E*. This comparison is done by taking the dot product of the represented value (from Equation 5) and each of the five target patterns.

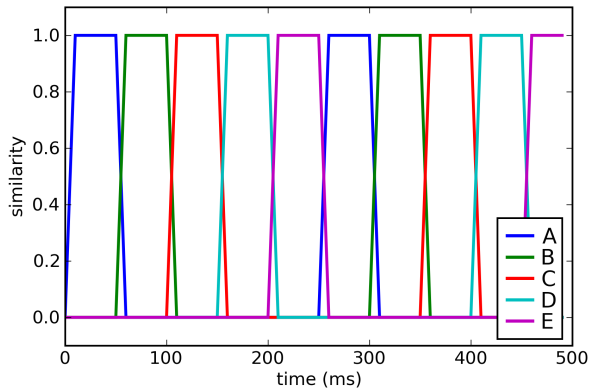

Figure 3: Behaviour of an ideal model cycling through five states, fixed to have a 50 millisecond cycle time.

Model 1: Basic Sequential Decisions

Our first model is created by adding a separate population of neurons for each of the rules to be implemented. These neurons must be connected to the main context neurons so that they will only fire when the value being represented is the same as (or very close to) the desired state (*A* through *E*). When a particular group of neurons starts to fire, their connections back to the context neurons are such that they will drive its firing towards the desired next state. This structure is shown in Figure 4. For clarity, this diagram shows only three rules: A→B, B→C, and C→A.

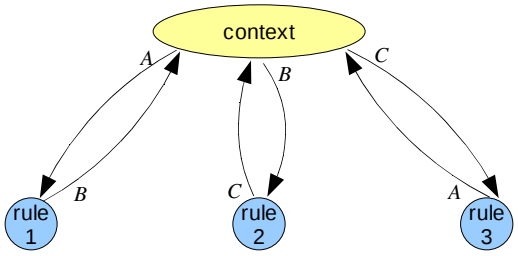

Figure 4: Neural groups and connections for Model 1.

To form the synaptic connections from the context to the rule neural groups, we can use Equation 6. For example, for the connection to the first rule, we set *M* to be the pattern *A*. As per Equation 6, this means that the neural group will be driven to represent the value $A\boldsymbol{x}$, which is the dot product of the represented context value with *A*. This will be large (near 1) when *A* is being represented, and small (near 0) when another pattern is being represented.

The properties of the neurons in the rule groups must also be set. Here, we can make use of the fact that we want these neurons to not fire at all when representing 0, but should be sensitive to values near 1. This can be achieved by having a large negative J_{bias} (with some random variation). The corresponding neuron tuning curves are shown in Figure 5. These show the average spiking rate of ten different neurons for different contexts $\boldsymbol{x}$. To see the actual spiking patterns over time, Figure 6 shows the spikes caused by varying the input to this neural group from 0 to 1 and back to 0 over one second.

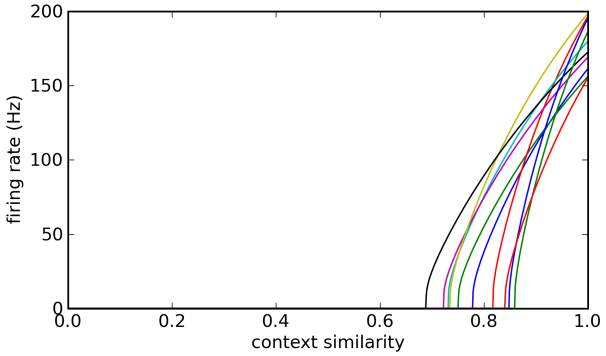

Figure 5: Average firing rates for neurons detecting the presence of pattern *A*. Different context patterns are on the x-axis: far left is a context unlike *A* (dot product of 0), far right is a context of exactly *A*. Each curve shows a different neuron with different values of α and J_{bias}.

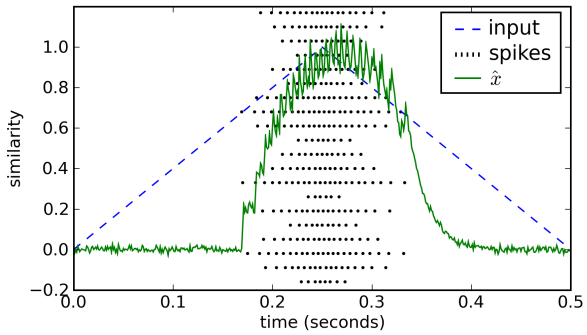

Figure 6: Individual neuron spikes for the neural group detecting *A*. Each neuron's spikes are separated along the y-axis. Dot product of the context with *A* is varied from 0 to 1 and back to 0 over one second (dotted line). The value $\hat{x}$ decoded using Equation 5 is shown (solid line).

We use a similar process to form synaptic connections from the individual rule groups back to the context neurons. Here, the weights encode the effect of each rule, indicating how the context should be changed if this rule is applied. These are again calculated using Equation 6. The resulting model has a variety of parameters, given in Table 1.

Table 1: Parameters of the model

Parameter	Default value
# of context neurons	2000
# of neurons per rule group	20
membrane time constant (τ_{RC})	20ms
synaptic time constant for context (τ_{SC})	10ms
synaptic time constant for rules (τ_{SR})	10ms

The behaviour of the resulting model is shown in Figure 7. As can be seen, it successfully cycles between the five states. For this particular model, each change requires an average of 27.5ms, making this much faster than the expected 50 millisecond cycle time. Furthermore, this rate is not sensitive to the numbers of neurons in each group: increasing these values by a factor of 10 causes only a slight decrease (<2ms) in the cycle time, since adding more neurons decreases the representational error in the system.

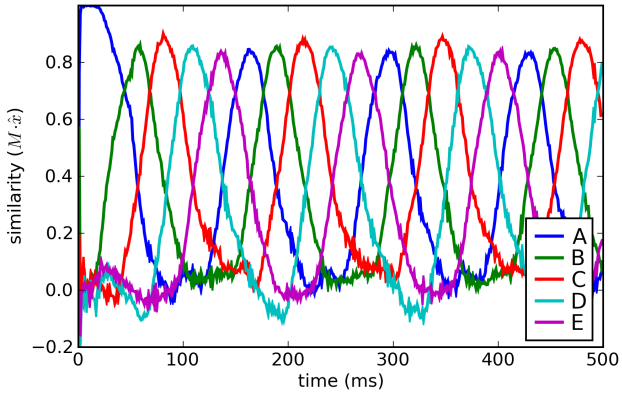

Figure 7: Behaviour of Model 1. Similarity is determined by the dot product of $\hat{x}$ (calculated from the spikes of the context neurons using Equation 5) with the vectors *A* to *E*.

The main effect on behaviour is seen by adjusting the synaptic time constants. As shown in Figure 2, different neurotransmitter/receptor pairs have different time constants. We can adjust the synapses from context neurons to rules separately from the ones from rule neurons to the context. These parameters are varied in Figure 8. The membrane time constant is known to be approximately 20ms for a wide range of neurons, so it is not adjusted here.

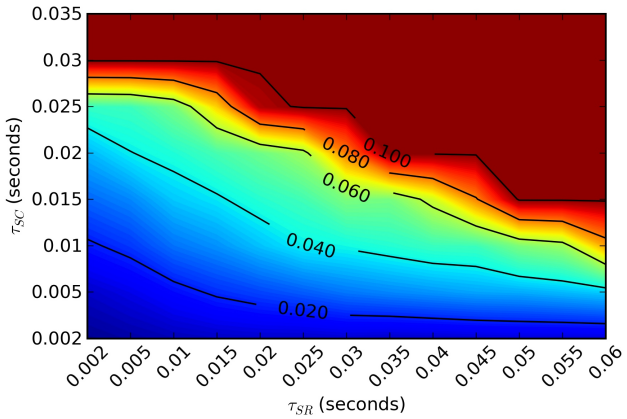

Figure 8: Average cycle time in seconds for varying τ_{SC} and τ_{SR} in Model 1. Values above 0.1 indicate either a cycle time above 100ms or no cycling (an infinite cycle time).

Given the results in Figure 8, the model is successful when the synaptic time constant for the context neurons is below 30ms, which is consistent with both GABA-A and fast AMPA synapses. This limit decreases as the synaptic time constant of the rule neurons increases.

While this model is successful at cycling across five different states, it fails in many other cases. For example, Figure 9 shows the behaviour when cycling between three states. Here, cycling behaviour is initially evident, but over time the system converges to a static representation. In particular, it converges to representing *all three* states at the same time. The final context value is the superposition (vector sum) of *A*, *B*, and *C*. This is clearly not the desired behaviour.

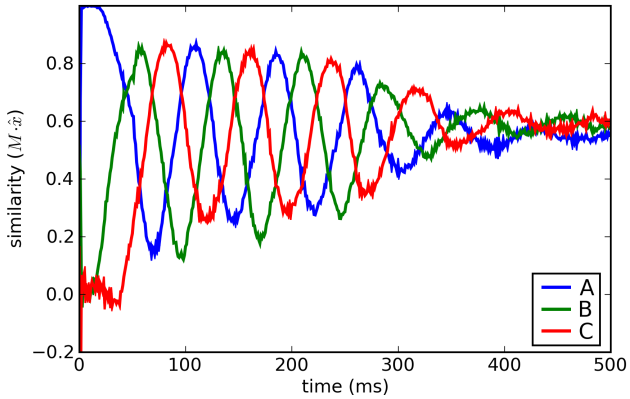

Figure 9: Behaviour of Model 1 when there are only three states. Similarity is determined by the dot product of $\hat{x}$ (calculated from the spikes of the context neurons using Equation 5) with the vectors *A*, *B*, and *C*.

Model 2: Inhibition Between Rules

To improve on Model 1 and fix the behaviour shown in Figure 9, we needed to add a mechanism to encourage the application of only one rule at a time. This was accomplished by adding inhibition between the groups of neurons responsible for each rule. That is, if the neurons in the first group are firing, this should decrease the activity in the other four groups. This is accomplished with Equation 6, where M is simply the value $-w_i$ (the strength of the inhibition). We must also add a self-excitatory connection of strength w_e within the neurons of each rule group, so as to counteract this inhibitory current. This new model is shown in Figure 10.

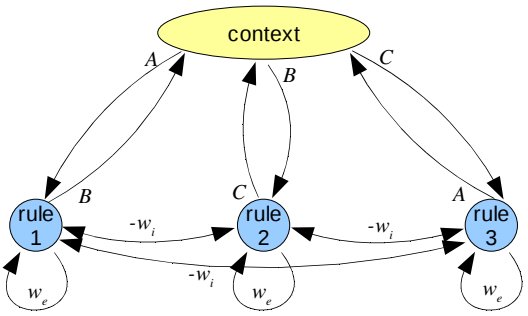

Figure 10: Neural groups and connections for Model 2.

For w_i of 0.5 and a w_e of 1, the model is successfully cyclic for cycles of 2 through 20 (which was as high as was tested). That is, the resulting behaviour looks like Figure 7, rather than Figure 9. The precise effects of these parameters will be explored in future work, as they are likely to impact any reinforcement learning system which might be used to bias one rule over another (such as in the ACT-R utility learning system). With these parameter values, the behaviour of the model for varying τ_{SC} and τ_{SR} is shown in Figure 11. We can see that Model 2 is slightly slower, but more stable over a wider range of synaptic time constants.

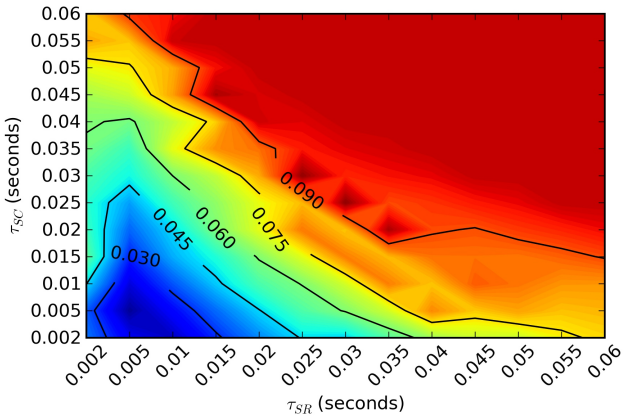

Figure 11: Average cycle time in seconds for varying τ_{SC} and τ_{SR} in Model 2. Values above 0.1 indicate either a cycle time above 100ms or no cycling (an infinite cycle time).

While this model eliminates the problem of convergence onto a superposition of states, there is a further difficulty present in both Model 1 and Model 2. So far, we have been assuming that this rule-following system is completely self-sufficient. In particular, once an action is chosen, the context is modified, and the system is then immediately able to start identifying the next rule to apply.

However, in real cognitive models, the central production system is only one of many components that can affect the current context. For example, in ACT-R, it is common for the production system to request that the declarative memory system recall a fact. While that fact is being recalled, the production system may not be doing anything, as no rules may apply until that fact is found (which may take hundreds of milliseconds). During that time, no rules are applied, but the context must be maintained.

Figure 12 shows the behaviour of Model 2 when no rules can be found that apply to the current context. This is done by removing the rule that transitions from E to A. As can be seen, when no rule can be applied, the system *forgets* the current context, since no rule is firing to set it in the context population. Model 1 behaves similarly.

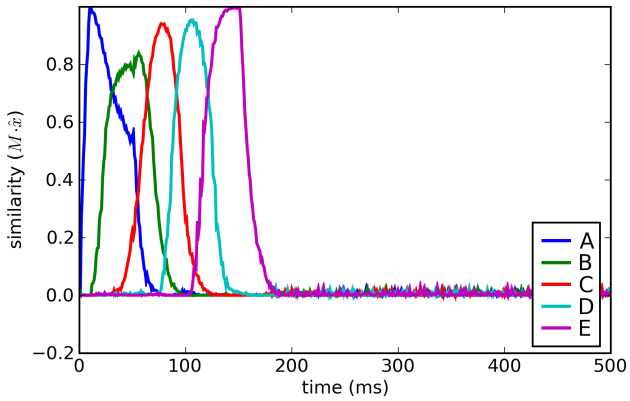

Figure 12: Behaviour of Model 2 when the rule to go from E to A is removed. The context information is lost.

Model 3: Maintaining Working Memory

To eliminate the forgetting effect shown in Figure 12, we add recurrent connections among the neurons representing context. This approach has previously been used to model working memory (Singh & Eliasmith, 2006), and is a generic method for storing information over time in spiking neurons. This is done by using Equation 6 to determine synaptic weights from the context population back into itself, with M set to be the identity matrix I. The resulting model is shown in Figure 13.

The behaviour of this model when the rule to transition from E to A is removed is shown in Figure 14. In contrast to Model 2 (Figure 12), the system is now capable of maintaining context information over time.

Adding this new recurrent connection allows information to be stored, but it also slows down the process of modifying this information. The behaviour for varying τ_{SC} and τ_{SR} is shown in Figure 15.

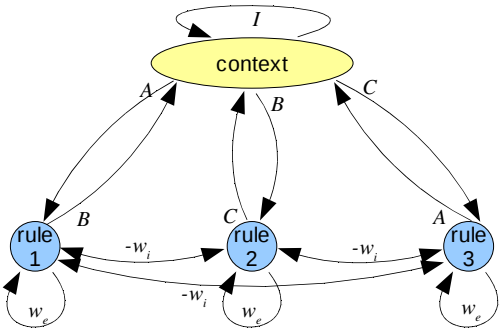

Figure 13: Neural groups and connections for Model 3.

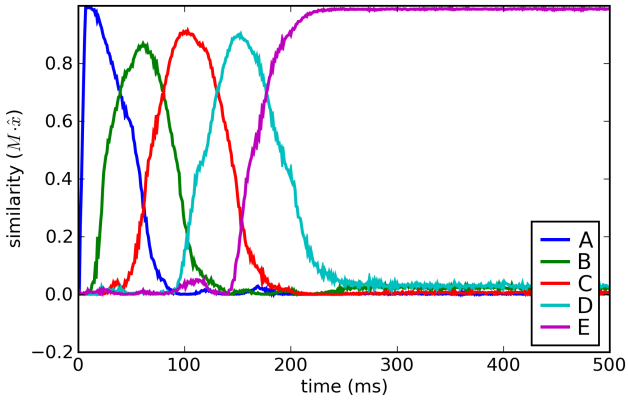

Figure 14: Behaviour of Model 3 when the rule to go from *E* to *A* is removed. The context information is maintained.

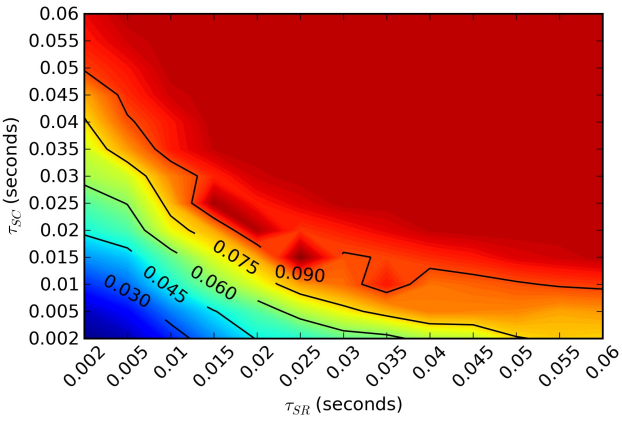

Figure 15: Average cycle time in seconds for varying τ_{SC} and τ_{SR} in Model 3. Values above 0.1 indicate either a cycle time above 100ms or no cycling (an infinite cycle time).

Discussion

Our Model 3 successfully identifies the rule appropriate to the current context and modifies the context appropriately. It is able to keep the patterns for each context separate (unlike Model 1) and store information over time (unlike Model 2). Furthermore, if the synaptic time constants for both the context neurons and the rule neurons are set to be 10ms, the average cycle time is 46.6ms, very close to the standard of 50ms. As noted above, 10ms is the synaptic time constant for GABA-A receptors. These are the primary synaptic receptors in the basal ganglia, which is the postulated location responsible for sequential rule selection.

While our model closely matches the generally accepted cycle time of 50 milliseconds, more is needed before it can be accepted as a neural model of central executive control. Most crucially, cognitive architectures generally postulate rules that are much more complex than "if A then B". We have shown elsewhere (Stewart & Eliasmith, 2008) how complex symbolic rules can be translated into vectors appropriate for our model. This would require the addition of a new neural population capable of combining the output of the rule neurons with the existing context. Preliminary results indicate that such a system would increase the cycle time by 5-10ms if AMPA or GABA-A are used.

We are also in the process of directly mapping our model onto the architecture of the basal ganglia and its connection to the cortex via the thalamus. In this case, the context may be stored using faster AMPA connections in various cortical areas and then gathered in the striatum for matching. The thalamus would then apply the complex rules mentioned in the previous paragraph. This is a direct match to the mapping from modules to brain areas found in ACT-R (Anderson et al., 2004). Furthermore, a learning system is required (likely using a dopamine-based expected reward signal) to identify how these synaptic connections arise.

Although our model is incomplete, it provides the first neural explanation for the 50 millisecond cognitive cycle. This time is a direct result of the properties of GABA-A receptors, along with the requirements that the system be able to recognize appropriate rules in a given context, apply rules separately, and store context information over time.

References

Anderson, J. R., John, B. E., Just, M. A., Carpenter, P. A., Kieras, D. E., & Meyer, D. E. (1995). *Production system models of complex cognition*. 17th Annual Meeting of the Cognitive Science Society.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review 111*(4). 1036-1060.

Eliasmith, C. & Anderson, C. (2003). *Neural Engineering: Computation, representation, and dynamics in neurobiological systems.* Cambridge: MIT Press.

Georgopoulos, A.P., Schwartz, A., & Kettner, R.E. (1986). Neuronal population coding of movement direction. *Science*, 233, 1416-1419.

Gupta, A., Wang, Y., & Markram, H. (2000). Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex. *Science*, 287. 273-278.

Koch, C. (1999). *Biophysics of computation: Information processing in single neurons.* Oxford University Press.

Moreno-Bote, R., & Parga, N. (2005). Simple model neurons with AMPA and NMDA filters: role of synaptic time scales. *Neurocomputing* 65-66. 441-448.

Singh, R., & Eliasmith, C. (2006). Higher-dimensional neurons explain the tuning and dynamics of working memory cells. *Journal of Neuroscience*, 26, 3667-3678.

Stewart, T.C. and Eliasmith, C. (2008). *Building production systems with realistic spiking neurons.* 30th Annual Meeting of the Cognitive Science Society.