ReScience

[Re] A neural model of the saccade generator in the reticular formation

Mario Senden[1, 2], **Jannis Schuecker**,[3], **Jan Hahne**,[4], **Markus Diesmann**,[3, 5, 6], **and Rainer Goebel**,[1, 2, 7]

1 Department of Cognitive Neuroscience, Faculty of Psychology and Neuroscience, Maastricht University, 6201BC Maastricht, The Netherlands **2** Maastricht Brain Imaging Centre, Faculty of Psychology and Neuroscience, Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands **3** Institute of Neuroscience and Medicine (INM-6) and Institute for Advanced Simulation (IAS-6) and JARA Institute Brain Structure-Function Relationships (INM-10), Jülich Research Centre, Jülich, Germany **4** School of Mathematics and Natural Sciences, Bergische Universität Wuppertal, Wuppertal, Germany **5** Department of Psychiatry, Psychotherapy and Psychosomatics, Medical Faculty, RWTH Aachen University, 52062 Aachen, Germany **6** Department of Physics, Faculty 1, RWTH Aachen University, 52062 Aachen, Germany **7** Department of Neuroimaging and Neuromodeling, Netherlands Institute for Neuroscience, an Institute of the Royal Netherlands Academy of Arts and Sciences (KNAW), 1105BA Amsterdam, The Netherlands

mario.senden@maastrichtuniversity.nl

Editor
Tiziano Zito

Reviewers
Andrew Davison
Georgios Is. Detorakis

Received February, 13, 2018
Accepted May, 4, 2018
Published May, 4, 2018

Competing Interests:
The authors have declared that no competing interests exist.

Article repository

Code repository

A reference implementation of

→ *A neural model of the saccade generator in the reticular formation*, G. Gancarz, S. Grossberg, Neural Networks, 1159-1174, 1998

Introduction

In 2013 the European Commission launched the Human Brain Project (HBP) tasked to build a research infrastructure spurring pan-European collaboration. Collaboration at such a scale, involving both empirically-oriented and theoretical neuroscientists, offers the unique opportunity to develop large-scale models of the brain. Specifically, individual research groups might develop models of distinct brain regions which can subsequently be combined into a unified whole. To facilitate this collaboration, the HBP encourages the utilization of publicly available, widely used, and actively developed neural simulation frameworks such as NEST [3]. In light of this, the NEST framework has recently been extended to support simulation of functionally inspired rate neuron models in addition to biologically grounded spiking neuron models [5]. Here we make use of this new functionality of the NEST framework (2.16.0) to provide an implementation of the saccade generator (SG); a rate neuron model of neural circuitry in the reticular formation proposed by Gancarz & Grossberg [2]. The SG is an integral part of the eye movement system [4] and as such vital for developing large-scale architectures of visuo-motor integration. We show that the model translates well to the NEST framework as our implementation faithfully reproduces all simulation results reported in the original publication. Our code uses the Python interface [1] for legibility with both model and analysis scripts being implemented using Python 2.7.12. However, the code is compatible with Python 3 as well (tested with version 3.5.2).

Methods

The SG model described by Gancarz & Grossberg [2] consists of a horizontal and a vertical component each with two long-lead burst neurons (LLBNs), excitatory burst neurons (EBNs), inhibitory burst neurons (IBNs), and tonic neurons (TNs). Within each component, the two directions (left-right, down-up) interact antagonistically. Additionally, both components share a single omnipause neuron (OPN) which tonically inhibits each EBN to suppress unwanted saccades. While this constitutes the core SG model, the superior colliculus (SC) is relevant for some simulations. In what follows we briefly describe the dynamics of neurons within the horizontal component (equivalent descriptions apply to the vertical component). Please note that subscripts u, d, l and r indicate whether a neuron controls upward, downward, leftward or rightward saccade direction, respectively.

Long-lead burst neurons

Each long-lead burst neuron (L) receives excitatory external input I and inhibitory feedback from the ipsilateral IBNs (B)

$$\begin{aligned} \tau \dot{L}_\text{l} &= -1.3L_\text{l} + I_\text{l} - 2B_\text{l} \\ \tau \dot{L}_\text{r} &= -1.3L_\text{r} + I_\text{r} - 2B_\text{r}. \end{aligned} \tag{1}$$

Excitatory burst neurons

Each excitatory burst neuron (E) receives excitatory input from the ipsilateral LLBN, a constant arousal signal equal to 1, and inhibitory input from the contralateral LLBN and the omnipause neuron (P)

$$\begin{aligned} \tau \dot{E}_\text{l} &= -3.5E_\text{l} + (2 - E_\text{l})(5L_\text{l} + 1) - (E_\text{l} + 1)(10L_\text{r} + 20g(P)) \\ \tau \dot{E}_\text{r} &= -3.5E_\text{r} + (2 - E_\text{r})(5L_\text{r} + 1) - (E_\text{r} + 1)(10L_\text{l} + 20g(P)), \end{aligned} \tag{2}$$

where the nonlinear gain function $g(\cdot)$ is given by

$$g(x) = \frac{x^4}{0.1^4 + x^4}. \tag{3}$$

Inhibitory burst neurons

Each inhibitory burst neuron (B) receives excitatory input from the ipsilateral EBN

$$\begin{aligned} \tau \dot{B}_\text{l} &= -2.4B_\text{l} + 3E_\text{l} \\ \tau \dot{B}_\text{r} &= -2.4B_\text{r} + 3E_\text{r}. \end{aligned} \tag{4}$$

Omnipause neuron

The omnipause neuron (P) receives inhibitory input from all LLBNs

$$\tau \dot{P} = -0.2P + (1 - P)(1.2 + J) - 3.5(P + 0.4)(g(L_\text{l}) + g(L_\text{r}) + g(L_\text{u}) + g(L_\text{d})), \tag{5}$$

where J represents external electrical stimulation and $g(\cdot)$ is again given by equation 3.

Tonic neurons

Each tonic neuron (T) receives excitatory input from the ipsilateral EBN and inhibitory input from the contralateral EBN

$$\begin{aligned} \tau \dot{T}_\mathrm{l} &= 0.1(E_\mathrm{l} - E_\mathrm{r}) \\ \tau \dot{T}_\mathrm{r} &= 0.1(E_\mathrm{r} - E_\mathrm{l}). \end{aligned} \tag{6}$$

The horizontal eye position (θ) depends on the activity of the right TN; i.e. $\theta = 260(T_\mathrm{r} - 0.5)$.

Superior colliculus

The superior colliculus (A) receives electrical stimulation F and evolves according to

$$\tau \dot{A} = -A + F. \tag{7}$$

SC input to the saccade generator

For a range of simulations, external input to the saccade generator is obtained by applying a nonlinearity $f(\cdot)$ given by

$$f(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ x & \text{if} \quad 0 < x < 1 \\ 1 & \text{if} \quad x \geq 1 \end{cases} \tag{8}$$

to the activity A of the SC and scaling the result by a weight W such that

$$I = W f(A). \tag{9}$$

In implementing this model, we largely follow the descriptions provided in the original publication with a number of motivated exceptions. Specifically, the original model description has two features which cannot be straightforwardly translated to NEST. First, a nonlinear gain function is applied to a subset of inputs to EBNs and the OPN while a linear gain function is applied to their remaining inputs. Since NEST applies the same gain function to all inputs of a neuron, we opt for using a linear gain function for EBNs and the OPN, but pass those inputs requiring an additional nonlinear gain function through an auxiliary unit instantaneously applying the desired nonlinearity before passing the result on to EBNs and the OPN. Second, constant input to a neuron is not hard-coded but rather provided by an appropriately weighted bias node. Neither of these changes leads to discrepancies with original results.

In all simulations we use the Exponential Euler (EE) method for numerical integration of all except tonic neurons [5] at a time step of 0.05 ms and a time constant of 50 ms. Tonic neurons are integrated using the Forward Euler method since they do not exhibit passive decay leading to division-by-zero when using the EE method. It should be noted that the initial firing rates of neurons are not in equilibrium. In order to address this, the model evolves for 100 ms to relax towards equilibrium before any input is applied. Furthermore, as in the original publication, any activity is bounded from below at zero. Finally, we always simulate the full model; i.e. both its horizontal and vertical components even if input is applied only to one of the two.

Results

All simulations of the original publication are repeated. Our results accord very well with those reported by Gancarz & Grossberg [2]. Only a single discrepancy is observed which we examine in depth.

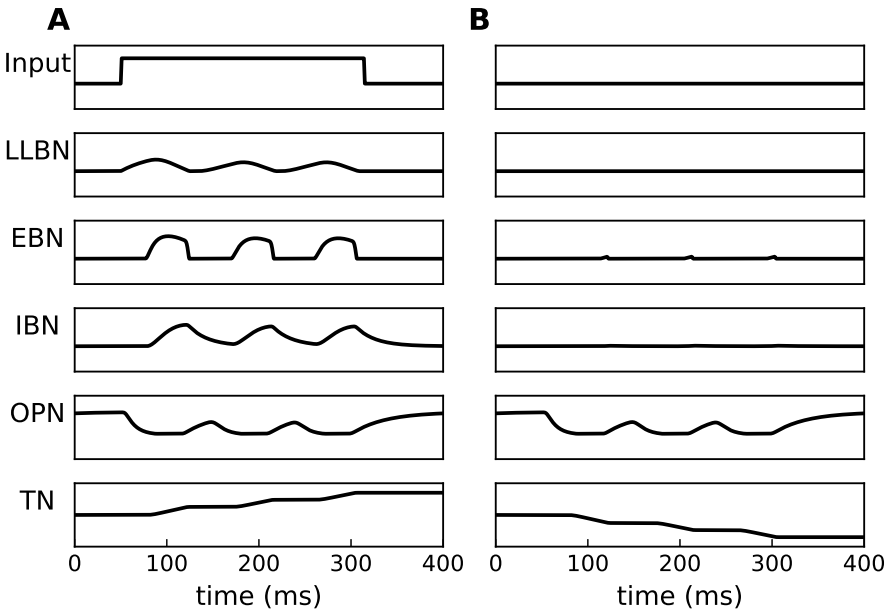

Figure 1: Activity profiles in the left (A) and right (B) SG. All activities are in response to constant input ($I = 1$) applied to the left LLBN for $265\,ms$.

The first simulation showcases the evolution of activity for each neuron type in the horizontal SG for a constant input applied to the left LLBN. The original publication does not report the exact activation values observed for each neuron rendering a quantitative analysis of the accuracy of our replication impossible. However, qualitatively activation profiles shown in figure 3 in the original publication and those shown in figure 1 show good correspondence. In both implementations, the left LLBN exhibits a prelude of activity with left EBN bursts beginning after the onset of LLBN activity. The right EBN produces a small burst at the end of a saccade. Furthermore, increases in activity of the left TN are mirrored by decreases in the right. Finally, OPN activity drops to zero during production of a saccade.

The second simulation shows the relation between input strength and burst amplitude for LLBNs and EBNs. With increasing input strength, both amplitude and duration of activation increase in long-lead and excitatory burst neurons. As before, these results accord well with those shown in figure 5 of the original publication.

The third simulation generates saccades in response to different input strengths applied to the horizontal and vertical SG (figure 3). As in the original publication, saccades are generally straight with a slight tendency to curve. Using PlotDigitizer, we extract saccade endpoints displayed in figure 6 of the original publication. This provides us with estimates of saccade amplitude allowing us to quantitatively asses our results in terms of the root-mean-squared error (RMSE). The RMSE was 0.16°, 0.17°, 0.18°, 0.22°, 0.14° for the blue, green, red, turquoise, and purple curves, respectively. Given that saccade amplitudes are between 10° and 15°, these RMSE values indicate a close match between the two implementations.

The fourth simulation implements a staircase of three saccades in response to continuous input (figure 4). Our results agree with those reported in the original publication (their figure 7) with saccades being of equal length and the smaller component (horizontal) being stretched to produce straight oblique saccades.

In the fifth simulation the average activity of the left EBN is obtained for a series of saccades with different directions. Figure 5 shows a polar plot of average activity corresponding to each saccade. This is the neuron's tuning curve. The tuning curve we observe for the left EBN exhibits a cardioid-like shape as in the original publication

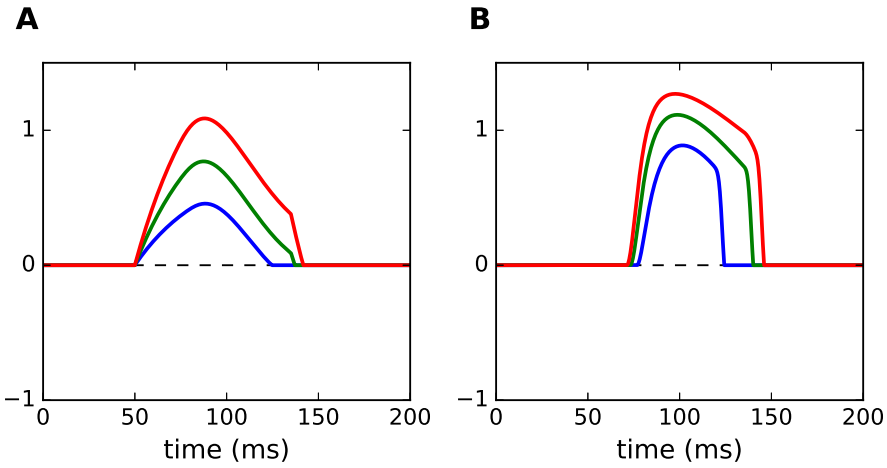

Figure 2: Activity profiles in LLBN (A) and EBN (B). Increased input strength resulted in larger LLBN and EBN burst size with inputs equal to 1 (blue), 1.75 (green), and 2.5 (red) each applied to the left LLBN for $85\,\mathrm{ms}$.

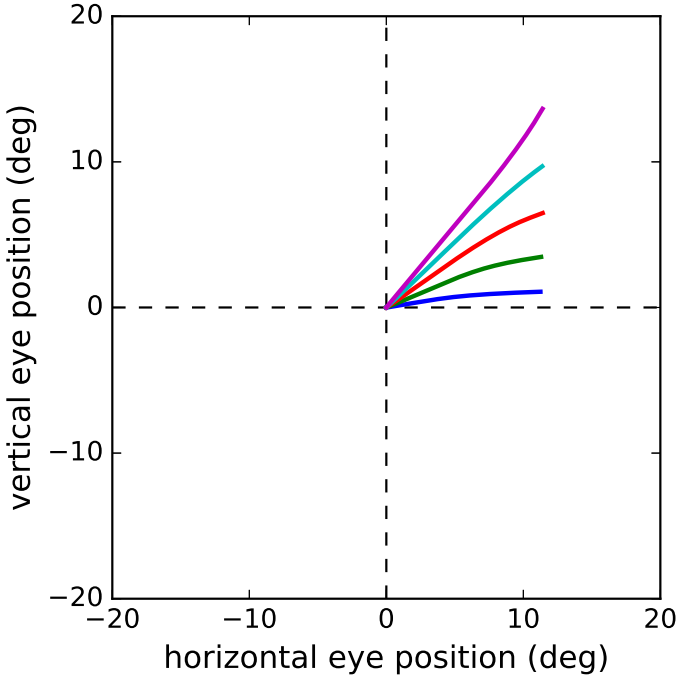

Figure 3: Oblique saccades. Inputs to the right and upward LLBNs are $\mathrm{I_r} = 0.67$ & $\mathrm{I_u} = 0.08$ (blue); $\mathrm{I_r} = 0.7$ & $\mathrm{I_u} = 0.22$ (green); $\mathrm{I_r} = 0.74$ & $\mathrm{I_u} = 0.4$ (red); $\mathrm{I_r} = 0.75$ & $\mathrm{I_u} = 0.6$ (turquoise); and $\mathrm{I_r} = 0.7$ & $\mathrm{I_u} = 0.9$ (purple) and are applied for $75\,\mathrm{ms}$.

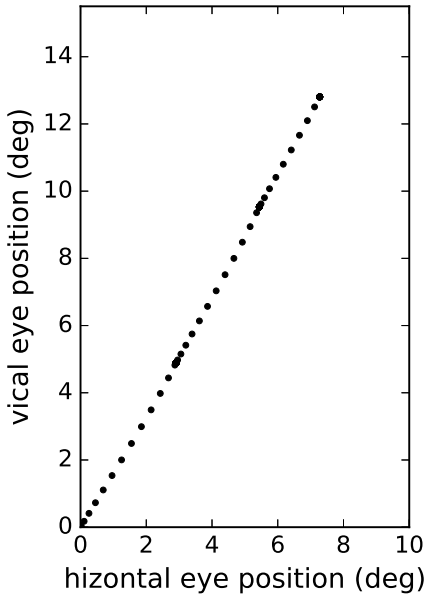

Figure 4: Saccades in a staircase. Eye positions exhibited in a saccade staircase simulation. Dense clusters reflect saccade endpoints. A total of three saccades are exhibited with subsequent saccades continuing in the same direction as the initial saccade. Inputs to the right and upward LLBNs are equal to $I_r = 0.2$ and $I_u = 0.33$ for 250 ms. The eye position is sampled every 2 ms.

(figure 8).

Table 1: Direction specific inputs to SG to produce EBN tuning curve.

	0	45	72	90	108	135	162	180	198	225	252	270	288	315
I_l	.00	.00	.00	.00	.20	.45	.63	.70	.63	.45	.20	.00	.00	.00
I_r	.70	.45	.20	.00	.00	.00	.00	.00	.00	.00	.00	.00	.20	.45
I_d	.00	.00	.00	.00	.00	.00	.00	.00	.20	.45	.63	.70	.63	.45
I_u	.00	.45	.63	.70	.63	.45	.20	.00	.00	.00	.00	.00	.00	.00

The sixth simulation reported in the original publication is designed to replicate results of Stanford *et al.* [6]. These authors stimulated the superior colliculus (SC) at various frequencies and measured the resulting saccade amplitude, duration, and velocity; showing that amplitude saturates before velocity. Our implementation of the SG model is capable of replicating these results. However, reproducing simulation results reported by Gancarz & Grossberg [2] with our implementation is complicated by the fact that the stimulation protocol given by the authors leads to the production of two rather than a single saccade for larger stimulation intensities. Furthermore, the authors do not report their criteria for identifying saccade on- and offsets. Stanford *et al.* [6] use velocity criteria to determine onset ($v > 30$ deg/s) and offset ($v < 30$ deg/s) of a saccade. While this provides us with explicit criteria, velocity does not always drop below 30 deg/s after the first saccade before rising again with the second. To determine the offset of a saccade in those cases, we find the local minimum between the end of the first and the beginning of the second saccade. With these criteria in place, we stimulate the SC. Results of our simulation are shown in figure 6. Again we use PlotDigitizer to extract a vector of saccade amplitude, duration and peak velocity observed at each stimulation frequency in the original publication from their figure 9 and calculate the RMSE between these and our curves. The RMSE is 1.24°, 0.46 ms, and 4.45 deg/s for saccade amplitude, duration, and peak velocity, respectively. Our results accord thus very well with those reported in the original publication.

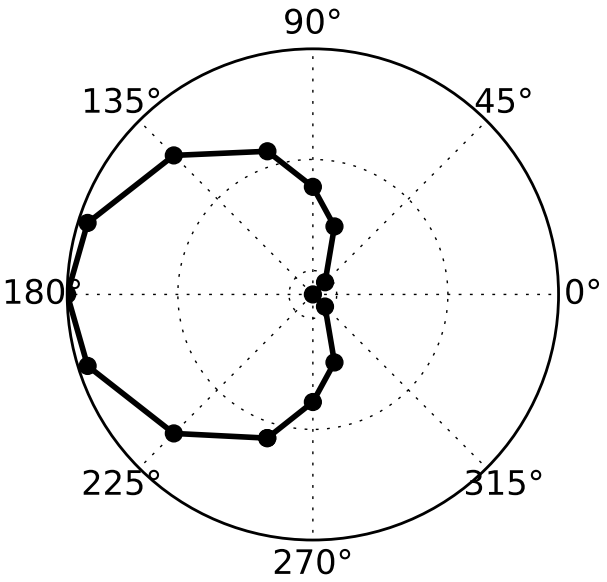

Figure 5: EBN tuning curve. Tuning curve of the left EBN exhibiting a cardioid-like shape. Inputs to the SG producing the saccades are given in table 1. Each of these inputs is applied for 50 ms.

This includes the observation that saccade amplitude and especially saccade duration are largest for a stimulation intensity of 1.2. It is conceivable that this intensity marks the point after which the SG starts producing two rather than a single saccade with the eye movement at this intensity merging two saccades and thus being stretched out.

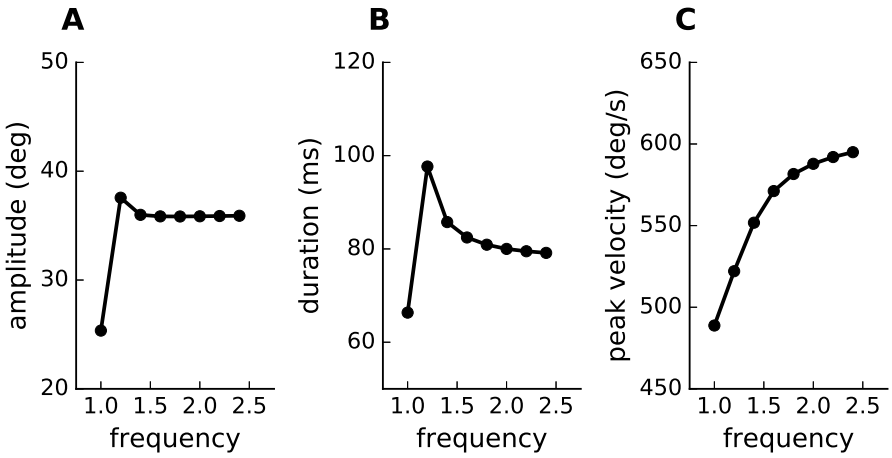

Figure 6: Effect of stimulation frequency on saccade amplitude (A), duration (B), and velocity (C). A range of unitless values varying between 1 and 2.4 at increments of 0.2, reflecting stimulation at different frequencies ,is applied to the SC. The connection weight from SC to LLBN is $\mathrm{W} = 2$ and stimulation duration is $125\,\mathrm{ms}$.

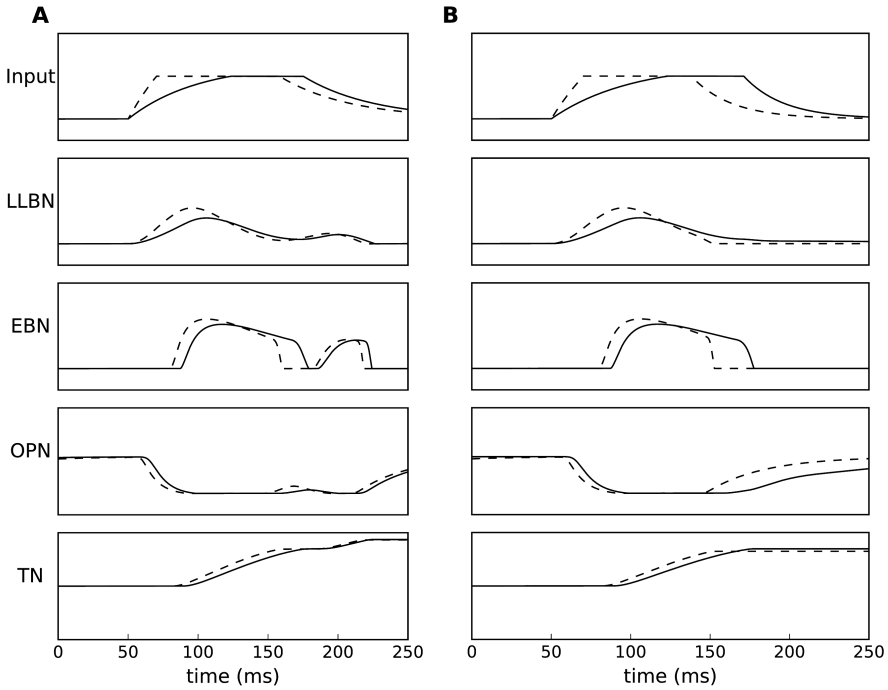

Figure 7: Trading saccade velocity for duration. Duration and velocity of a saccade can be traded while keeping amplitude constant. To produce a high velicity saccade an input of $\mathrm{F} = 3$ was applied to the SC for $68\,\mathrm{ms}$ ($82\,\mathrm{ms}$ in the original publication; solid curve). To produce a low velocity saccade an input of $\mathrm{F} = 1.3$ for $117\,\mathrm{ms}$ (dashed curve). Given the shape of the input curve produced by our implementation of the original equations (A), a second saccade can be observed for both high and low velocity saccades. If the input curves reported in the original manuscript are recreated in an ad-hoc fashion (B), the responses of all model neurons matches those in [2]

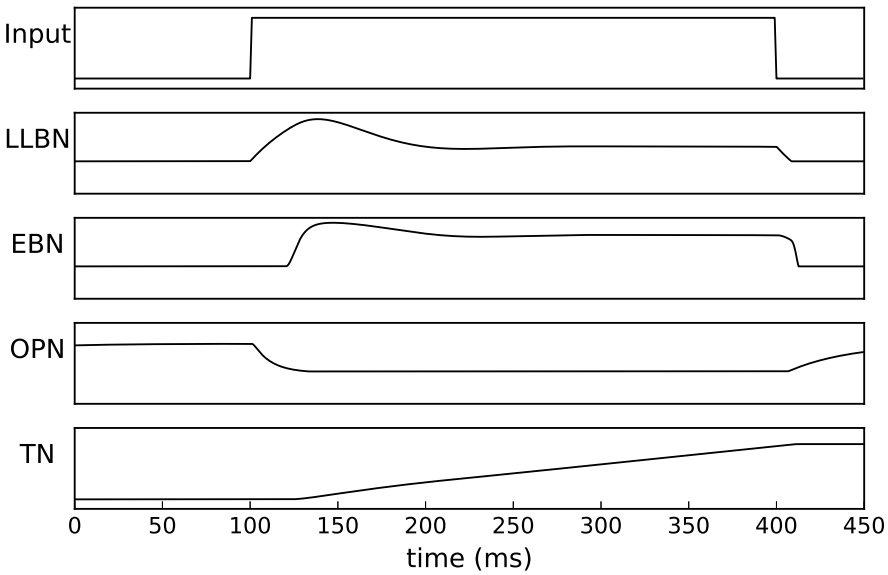

Figure 8: Smooth staircase eye movements. Activity profiles of SG neurons accompanying smooth eyes movement as a result of strong ($I = 3$) sustained ($300\,\mathrm{ms}$) input.

The seventh simulation shows that saccade velocity and duration can be traded while keeping amplitude constant. To produce a high-velocity saccade, the SC is stimulated at a high frequency. Conversely, to produce a low-velocity saccade, the SC is stimulated at a low frequency. Figure 7 shows the results of our simulation. In line with the original publication, the saccade amplitude reflected by TN activity is identical after high- and low-frequency saccades thus confirming the reported effect. However, we observe differences in the specific details of our implementation with respect to the original, starting with the shape of the input curves. We evaluate rise and decay of input curves reported in the original publication by estimating their onset as well as time constants from points obtained from figure 10 using PlotDigitizer. While the rise of the input curves in the original publication and our implementation coincide well, the decay in the original curves starts later and proceeds with a time constant of about half the magnitude in our simulation ($\approx 25\,\mathrm{ms}$). This results in more total input to the model in our implementation and hence leads to the generation of a second saccade not observed in the original publication. When repeating our simulation using an ad-hoc solution to create matching input curves by halving the time constant after external stimulation, we observe results identical to those reported in the original publication. Next we investigate the origin of the discrepancy with regard to the shapes of the input curves. First, we test whether discrepancies are specific to our NEST implementation by solving the expressions describing the input analytically. Specifically, the analytic expression of the input $I(t)$ described by equations 7 - 9 is

$$I(t) = \begin{cases} 0 & \text{if} \quad t \leq t_{on} \\ W \cdot f(F \cdot (1 - e^{-(t-t_{on})/\tau})) & \text{if} \quad t_{on} < t < t_{off} \\ W \cdot f(F \cdot (1 - e^{-(t_{off}-t_{on})/\tau}) \cdot e^{-(t-t_{off})/\tau}) & \text{if} \quad t \geq t_{off} \end{cases}. \tag{10}$$

Evaluating the analytic expression given parameter values reported in the original manuscript exactly reproduces our numerical results and thus produces curves equally deviating from those shown in the original publication. At this point we contacted one of the original authors to rule out the possibility that we use erroneous parameter settings but no mistake was found. While the precise origin of the discrepancy thus remains unclear, most likely the original code uses different time constants for rise and decay.

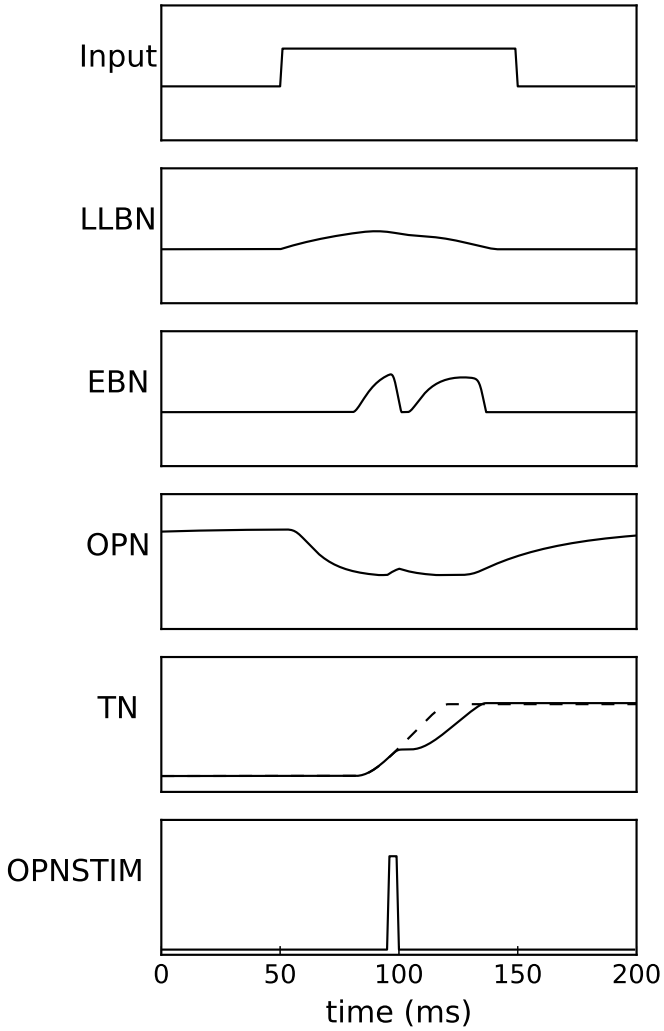

Figure 9: Interrupted saccade resulting from OPN stimulation. OPN stimulation interrupts the saccade, which remains accurate nonetheless. An input of ($\mathrm{I} = 0.7$) is applied to the LLBN for $100\,\mathrm{ms}$. At $45\,\mathrm{ms}$ after onset of the input, the OPN is stimulated ($\mathrm{J} = 1.8$) for $5\,\mathrm{ms}$. The dashed curve shows TN activity for an uninterrupted saccade.

The eighth simulation reported by Gancarz & Grossberg [2] shows how strong sustained input to the SG produces smooth eye movements. Our results, shown in figure 8, reproduce these findings as they strongly resemble those shown in figure 11 of the original publication. Specifically, an initial burst exhibited by the EBN is followed by sustained lower activity. This is due to inhibitory feedback being insufficient to silence the LLBN when input remains continuously strong and results, in turn, in the observed smooth eye movement.

The final simulation showcases the evolution of activity exhibited by SG neurons when the OPN is briefly electrically stimulated while a constant input is applied to the LLBN. External stimulation temporarily restores activation in the OPN and hence also inhibition of the EBN, leading to an interruption of the saccade. As is shown in figure 9, the saccade remains accurate despite this disruption. This is in agreement with results shown in figure 12 of the original publication.

Conclusion

The reproduced results show very good qualitative correspondence with those reported by Gancarz and Grossberg [2] and accord well quantitatively wherever such information is available. The only discrepancy between our and the original implementation is that our implementation produces a second saccade in simulation 7 not observed in the original publication. While the exact cause of this discrepancy remains unclear, we could trace it back to the shape of the input curves. An analytical treatment reveals that the discrepant curves result from the description of the input allowing us to rule out that they originate from the NEST implementation. Furthermore, when the SG model is provided with corrected input curves, it faithfully reproduces the results reported in the original publication for simulation 7; namely that saccade velocity and duration can be traded while keeping the amplitude constant. There thus appear to be no issues with either the NEST implementation nor the SG model.

In conclusion, our reproduction confirms the results of the original publication and shows that an implementation in the NEST framework is feasible. This allows for the straightforward integration of the saccade generator with computational models of other components of the visuo-motor system (e.g. salience computation) within a shared framework.

Acknowledgments

All network simulations carried out with NEST (http://www.nest-simulator.org). This research was funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreements No. 7202070 (HBP SGA1) and 737691 (HBP SGA2).

Appendix

A1 - Rate neurons in NEST

For a detailed, comprehensive, treatment of rate-based neuron models in NEST see Hahne et al. [5]. Here we provide only a brief overview to facilitate readability and interpretability of the code provided with this publication. In general, rate-based model neurons in NEST consist of two components; an abstract neuron class and a gain function.

Neuron class

Two types of neuron base classes exist in NEST 2.16.0: `rate_neuron` and `rate_transformer_node`. The `rate_neuron` implements continuous rate dynamics with and without multiplicative coupling. It can apply a nonlinear gain function to its inputs either before or after their weighted summation. The model is numerically integrated using the EE method unless a neuron does not exhibit exponential decay in which case the Euler-Maruyama method is used. Furthermore, rates can be bounded (from below at zero) or unbounded. Finally, noise (in the form of a Wiener process) can be added to its input or to its output. The `rate_neuron` has a total of seven parameters (see table 2) and two further properties `rate` (current firing rate) and `noise` (current noise level).

Table 2: `rate_neuron` parameters.

parameter	default value	description
`tau`	10 ms	time constant
`lambda`	1	passive decay rate
`std`	1	noise standard deviation
`mean`	0	mean firing rate
`linear_summation`	`true`	if true, apply gain function after summation
`rectify_output`	`false`	if true, rate is bounded
`mult_coupling`	`false`	if true, multiplicative coupling

For the implementation of this neuron see `rate_neuron_ipn_impl.h` and `rate_neuron_opn_impl.h` in `../nest/nest-simulator-master/models` for input and output noise, respectively.

The `rate_transformer_node` instantaneously applies a nonlinearity to its inputs but does not exhibit temporal dynamics. It can be used to apply different nonlinearities to different inputs to a single neuron. This class has only the `rate` property and no parameters. For its implementation see `rate_transformer_node_impl.h` in `../nest/nest-simulator-master/models`.

Gain function

Five gain functions exist in NEST 2.16.0: `lin_rate`, `sigmoid_rate`, `sigmoid_rate_gg_1998`, `tanh_rate`, and `threshold_lin_rate`. Documentation of each gain function is given in the header file with the corresponding name within `../nest/nest-simulator-master/models`. We added `sigmoid_rate_gg_1998` specifically for the reimplementation of the saccade generator presented here.

Gain functions are combined with a neuron class to obtain a specific neuron model. For instance, the combination of the `lin_rate` gain function with the `rate_neuron_ipn` class gives the `lin_rate_ipn` neuron model. The parameters of a model are a combination of all parameters of the neuron base class and all parameters of the gain function. Within Python the command `nest.Models()` lists all existing model types including all implemented combinations of gain function and neuron class.

References

[1] Jochen M Eppler. "PyNEST: A convenient interface to the NEST simulator". In: *Frontiers in Neuroinformatics* 2 (2008). ISSN: 16625196. DOI: 10.3389/neuro.11.012.2008.

[2] Gregory Gancarz and Stephen Grossberg. "A neural model of the saccade generator in the reticular formation". In: *Neural Networks* 11.7 (1998), pp. 1159–1174. ISSN: 08936080. DOI: 10.1016/S0893-6080(98)00096-3.

[3] Marc-Oliver Gewaltig and Markus Diesmann. "NEST (Neural Simulation Tool)". In: *Scholarpedia* 2.4 (2007), p. 1430. ISSN: 1941-6016. DOI: 10.4249/scholarpedia.1430.

[4] Stephen Grossberg, Krishna Srihasam, and Daniel Bullock. "Neural dynamics of saccadic and smooth pursuit eye movement coordination during visual tracking of unpredictably moving targets". In: *Neural Networks* 27 (2012), pp. 1–20. ISSN: 08936080. DOI: 10.1016/j.neunet.2011.10.011.

[5] Jan Hahne et al. "Integration of Continuous-Time Dynamics in a Spiking Neural Network Simulator". In: *Frontiers in Neuroinformatics* 11 (May 2017), p. 34. ISSN: 1662-5196. DOI: 10.3389/fninf.2017.00034.

[6] T R Stanford, E G Freedman, and D L Sparks. "Site and parameters of microstimulation: evidence for independent effects on the properties of saccades evoked from the primate superior colliculus." In: *Journal of neurophysiology* 76.5 (Nov. 1996), pp. 3360–81. ISSN: 0022-3077.