

[Rp] Dendrodendritic inhibition and simulated odor responses in a detailed olfactory bulb network model

Andrew P. Davison

▶ **To cite this version:**

Andrew P. Davison. [Rp] Dendrodendritic inhibition and simulated odor responses in a detailed olfactory bulb network model. ReScience C, 2020, 10.5281/zenodo.3972130 . hal-03046451

HAL Id: hal-03046451
https://hal.science/hal-03046451

Submitted on 8 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RESCIENCE C

Reproduction / Computational Neuroscience

[Rp] Dendrodendritic inhibition and simulated odor responses in a detailed olfactory bulb network model

Andrew P. Davison[1, ID]

[1]Université Paris-Saclay, CNRS, Institut des neurosciences Paris-Saclay, 91190, Gif-sur-Yvette, France

Edited by
Tiziano Zito

Reviewed by
Nicolas P. Rougier [ID]

Received
30 April 2020

Published
05 August 2020

DOI
10.5281/zenodo.3972130

This article reports on the experience of reproducing a computational modelling study I originally carried out as part of my PhD studies, which was published as Davison, Feng, and Brown (2003)[1].

1 Historical context

1.1 Scientific context

The code was written to explore the question of how odour information is processed by the olfactory bulb. At the time it was impossible to measure simultaneously the spatial distribution and the temporal structure of neuronal activity in the bulb, so our goal was to develop a biologically 'realistic' model [1], validate it with respect to available data, then use it to explore properties of network activity in response to different forms of stimulation. The resulting model was the most realistic, largest-scale model published at that time, although it has subsequently been surpassed in both these dimensions, of course (see [2], for example).

1.2 Computational context

The model consisted of compartmental models of mitral and granule cells, connected by phenomenological models of AMPA, NMDA and $GABA_A$ synapses. In computational neuroscience, compartmental modelling represents the spatially extended structure of neurons as branching electrical cables, then spatially discretizes the cable equation into isopotential compartments [3]. Compartmental models of the mitral and granule cells of the olfactory bulb had been developed previously by Bhalla and Bower (1993)[4] for the GENESIS simulator[5], and I intended to build on this basis. However, although I was able to compile GENESIS itself on the DEC Alpha workstation which was the main computational workhorse of the lab, I was unsuccessful in compiling its associated graphical user interface (XODUS), probably due to my very limited computational training (self-taught BASIC as a child, then an undergraduate class in FORTRAN for physicists and a graduate course in image processing using MATLAB). I therefore decided to attempt to port the models to different simulation software, NEURON [6], which I was able to compile in its entirety, GUI and all. This process took three months, the first month on the beautiful island of Crete, at the European computational neuroscience

[1]as realistic as possible given the limitations of scientific knowledge about the olfactory bulb and the limitations of available computing resources at the time

Copyright © 2020 A.P. Davison, released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to Andrew P. Davison (andrew.davison@cnrs.fr)
The authors have declared that no competing interests exist.
Code is available at https://github.com/apdavison/bulbnet-reproduction. – SWH swh:1:rev:0899bc8af36b50d8cdf5d062562d2721f4602703.
Open peer review is available at https://github.com/ReScience/submissions/issues/37.

summer school, held at the FORTH institute in Heraklion where I could benefit from hands-on tutoring from Michael Hines, principal author of NEURON. The advantage of the porting process was that it required me to learn about both NEURON and GENESIS in considerable depth, as well as requiring me to understand more general theory about numerical solutions of differential equations. Finally, I was able to achieve near identical results from the two implementations (to better than nine significant figures, if I remember correctly), although the voltage-gated calcium channels were a particular challenge.

The next problem was that the neuron models, with several hundred compartments, each containing a half-dozen ion channels or more, each of these with 3-4 coupled, stiff differential equations to solve (Hodgkin-Huxley formalism), were computationally very expensive, and it was not possible to simulate very many of them at the same time, as was needed for a network model of the bulb. I therefore investigated the possibility of simplifying the geometric complexity of the neuron models, and found that a mitral cell model reduced to four compartments and a granule cell model with three compartments gave adequate results [7].

Models in the NEURON simulation environment were written in two languages: NMODL for ion channel and synapse models [8], a declarative specification that is used to generate C-code, which is then compiled and linked to the main NEURON executable; and Hoc [9] for creating neuron models, linking them in networks, and general programming (Hoc was my first introduction to object-oriented programming) [2]. Figures were mostly generated using Gnuplot (http://www.gnuplot.info/); Figures 15 and 16 were hand-written Postscript. Some data analysis was done interactively at the command line using awk, sed and other UNIX command-line tools. The NMODL files tended to undergo little or no change after they were first developed. However, each new experiment resulted in copying, renaming and modifying multiple Hoc files. Essentially, every figure in the article was associated with a collection of Hoc files, with a core of code that was copied mostly unchanged from file to file, but considerable additional or ancillary code at each iteration.

1.3 Reproducibility and replicability

At this point, the head of the lab, David Brown, expressed concern about the accuracy of the NEURON simulator. NEURON was a complex piece of software. How could I be sure that it was solving the differential equations correctly and with a sufficient level of accuracy? To address this concern, I reimplemented the four-compartment mitral cell model using the XPPAUT differential equation solver [10] and obtained essentially identical results (again, to multiple significant figures). This experience, of porting models between three different simulators, impressed upon me the importance of cross-validating simulation results across different implementations, and of the importance of checking the accuracy of numerical solutions, for example by reducing the integration time step until the results converge.

It might be asked, why use simulation software at all, rather than write code from scratch, or based on lower-level libraries? It became clear to me during this experience that there were two principal benefits. The first is increased productivity; since the programming environment for the simulator expresses domain-specific concepts, models can be developed more rapidly, without having to worry too much about low-level details. The second benefit is reduced bugs; given the complexity of the domain that is being modelled, combining lower-level libraries and writing large amounts of code inevitably leads to bugs, and often results in hidden conceptual errors. It is usually better to use software that has been used and tested by many people over many years. This may seem obvious today, but the argument that it is essential to understand every detail of your code down

[2] NEURON now allows the use of Python in the place of Hoc

to the compiler, and that the best way to have this understanding is to write the code yourself, still carries considerable weight.
As mentioned previously, I had little programming experience before starting my PhD studies, and no training in formal software development. At the beginning I was not aware of the existence of version control software. At some later point I encountered RCS and CVS, but they were not compelling enough for me to adopt them. It was not until Subversion came along a couple of years later that I started using formal version control. During this project, then, I used the time-honoured version control method of copying the file and changing the file name.
In summary, although reproducibility was not discussed with me by my supervisors as such, it was implicitly seen as an essential part of general scientific rigour. Similarly, although I did not explicitly have the idea that the re-usability of the code I was writing was an important factor, I was building on previously published models and, perhaps more importantly, previously published code, since the GENESIS code for the mitral and granule cells was distributed through BABEL, the GENESIS users' group, and without access to this code I could never have been certain that my NEURON implementation was really identical. I therefore implicitly had the expectation that the code I wrote might be used by others in future.

1.4 Original source code

Around the time that the article was nearing publication (2-3 years after submitting my thesis!), I decided to deposit a version of the code with the ModelDB repository [11], accession number 2730. To strike a balance between promoting reproducibility and not discouraging submissions, the policy of ModelDB is that the deposited code must reproduce at least one of the figures from the associated article, but not necessarily all of them. I therefore cleaned up and refactored the code from my thesis into a smaller number of files, with the idea that different experiments could be performed by changing a parameter/configuration file, but without needing to change the Hoc or NMODL code itself. The version of the code deposited with ModelDB reproduced Figures 2 and 8 from the article. The goal of this current project was to reproduce the other thirteen data figures.
The code on ModelDB has no explicit licence, in common with the majority of ModelDB entries. This general lack of licenses does not appear to inhibit researchers in computational neuroscience from making use of code in ModelDB, nor from publishing new studies based on ModelDB code. Since every model in ModelDB is associated with a published article or book chapter, the reuse of code is acknowledged primarily through academic citations.

2 Retrieval of the software

As noted above, a version of the source code that reproduced two figures was available on ModelDB (it is perhaps impressive that ModelDB still exists, as many neuroinformatics resources funded by the US National Institutes of Health (NIH) 'Human Brain Project'[3] in the 1990s are no longer available online.) This version of the code was not in itself sufficient to reproduce the other thirteen figures.
The original source code from my thesis, and from additional simulations requested by reviewers during the article publication process, had been copied between five or six computers in three countries over two continents, and reorganised on several occasions. Two of these computers were sitting in my office, but had not been turned on in many years. There was no copy of the code on either of my current computers. I had also made regular backups of key folders onto CD-R and CD-RW optical disks, and had several

[3] Not to be confused with the current EU research project of the same name

dozen of these backups to hand. The lifespan of such disks should be at least ten years (100 years or more for CD-R, 25 years or more for CD-RW, according to https://www.clir.org/pubs/reports/pub121/sec4/). Finding a drive able to read them was potentially more of a problem, but one of my machines still has a CD/DVD drive, and after blowing out the dust it was able to read all the 20 or so disks I tried.
On these backups, I found multiple copies of the code for almost all the figures, although not, for some reason, for Figures 15 and 16 (see below).
The source code for all versions of NEURON back to the year 1992 is available from the NEURON website. Gnuplot, used for creating figures, is available as a Debian package. Older versions back to the year 2000 are available on Sourceforge.

3 Execution

3.1 Computational environment

I do not recall precisely which version(s) of NEURON was/were used for the simulations shown in the original article. Based on the timing of the work, it was probably NEURON 4.3. I considered trying to install this version on a modern machine, but it seemed simpler to first try the current release of NEURON, since the software has been continuously maintained and developed by a stable development team (in particular Dr Michael Hines), with a reputation for not breaking backwards compatibility. Similarly, I most probably used Gnuplot version 3.7.1 at the time, but used the most recent version for this reproduction attempt.
The simulations for this study were performed on a Dell Precision 7910 workstation, with dual Intel Xeon E5-2640 processors (8 cores, 2.6 GHz, 20 MB cache), 128 GB 2400 MHz DDR4 ECC RAM, writing to a 400 GB SSD SAS 12 Gbit/s hard disk. The operating system was Ubuntu 18.04. The software used was: NEURON v7.6.7, compiled with gcc 7.4.0; Python 3.8.1.; Numpy 1.18.1; PyYAML 5.3; Jinja 2.11.1; and Gnuplot 5.2.

3.2 Code modifications

All the NEURON code ran without modification, although I was able to benefit from the support for multithreading, introduced since the original study, to obtain a speedup of 8 times when running with 32 threads (16 cores, presumably using hyperthreading). My memory is that a typical simulation took about six hours on the DEC Alpha workstation I used in 1998-2000. The shortest simulations in my current environment take 11 minutes single-threaded and 80 seconds using 32 threads. The Gnuplot scripts required very little modification; at some point the Gnuplot commands were revised for consistency, so for example '`set data style lines`' became '`set style data lines`'.

My goal for this reproduction was to fully automate the process of running simulations and generating publication-ready figures, with none of the manual copy-pasting, script editing and command-line data analysis that was necessary originally. To this end, I firstly completed the task of fully parameterizing the Hoc code, so each experiment corresponds to a particular set of parameters, but no code changes are needed for different experiments. I secondly wrote a Python script to automate everything. Each data figure in the original article corresponds to a function in this script. Most of the work is still done by NEURON and Gnuplot, but intermediate data formatting and analysis is done in Python. The hand-written Postscript figures have been turned into Jinja templates, which are filled with data coming from the simulations.

3.3 Obtaining and running the code

All of the code for the reproduction is available at https://github.com/apdavison/bulbnet-reproduction under an MIT Licence.
To reproduce Figure 11, say, on the command line run:

```
$ python run.py figure_11
```

and similarly for the other figures (2-5 and 7-17; Figure 1 was an illustration of the network structure, Figure 6 an illustration of a proposed mechanism).

4 Results

For none of the figures were we able to exactly reproduce the originally published ones, due to differences in the sequences of random numbers. However, all figures, with the exception of Figure 13 (Figure 11 in this article) gave qualitatively the same result, and were quantitatively very close. Responses of individual cells, particularly granule cells (Figure 1B) which receive monosynaptic excitation, can be very different from the original, as these responses depend strongly on the number of connections a cell receives, and that depends on the random number generator seed. However, measures of population activity, for example the population spike time histogram in Figure 1E, or even individual neuron responses that depend on disynaptic connections, such as the mitral cell inhibitory post-synaptic current (IPSC; Figure 1C), are very well preserved in the reproduction. Figure 1F shows that the mitral cell IPSC from the original figure lies within the range of results obtained with different random number generator seeds.
While it is disappointing that we couldn't obtain quantitatively identical results, the robustness of the scientific conclusions to differences in random number sequences is a very positive property of the model.

5 Conclusions

Could another researcher have reproduced the original results? For the two figures that were reproduced by the code deposited at ModelDB, the answer is yes. The code works with the current version of NEURON, and the README contains clear and simple instructions.
For the other figures, with access to the backups of my working files, the answer is still yes, although with more difficulty, since some 'detective' work is required in matching the names of Hoc files to the corresponding figure.
Without these backups, but with the ModelDB version, it would still be possible to reproduce the results, but it would take several days, probably a couple of weeks. The caption of each of the figures in the original article contains a list of the model parameters that were different from the defaults, and the defaults are given in the Methods section and in a table. However, some parameters are missing from the article, in particular the parameters of the background input for Figures 15 and 16, and some trial-and-error would be required to recover parameters similar to those used originally.
In summary, I regard this as a successful reproduction. In retrospect it would have been better to deposit code for all figures in ModelDB, not just for two of them, at a time when the details of the code were much fresher in my memory. It would also have been valuable to go through the source code and systematically check that all parameters and input data were described in the published article; most of them were, but some are missing.
The choice of using well-maintained, stable, widely-used software was a very good one. NEURON required no code modifications (although code could be added to take advan-

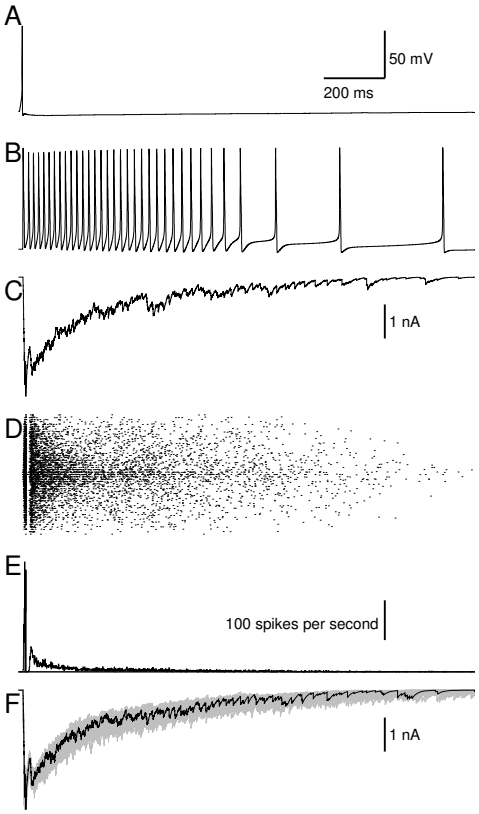


Figure 1. Reproduction of Figure 2. The precise number and timing of spikes in the example neuron is different, due to the random element in connecting the networks, but the shape of the IPSC and the spatial and temporal profiles of the population activity are very similar. Panel **F** is an addition: it shows the IPSC for 50 simulations differing only in the seed to the random number generator (grey), overlaid with the IPSC from the original Figure 2.

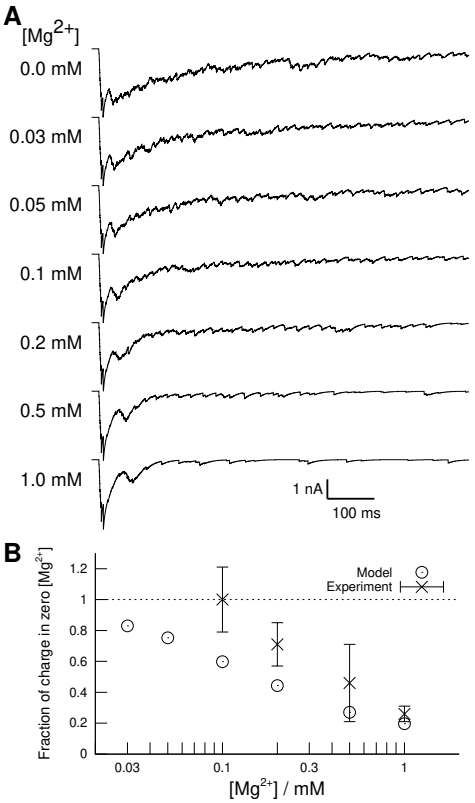


Figure 2. Reproduction of Figure 3. The results are visually identical to the original.

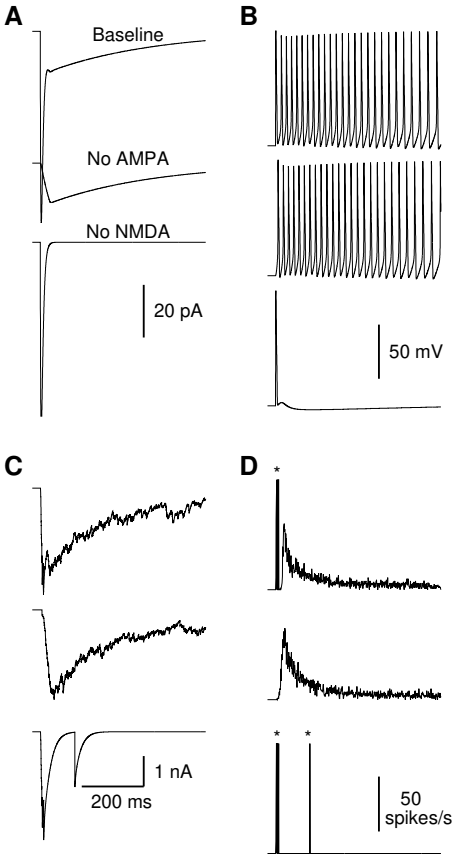


Figure 3. Reproduction of Figure 4. Due to the different, random, connectivity in the reproduction, the individual granule cells show differences in their responses (A, B); however at the population level (D), and even at the level of the di-synaptic mitral cell response (C), the results are very close to the original, with differences in the fine details of the noise.

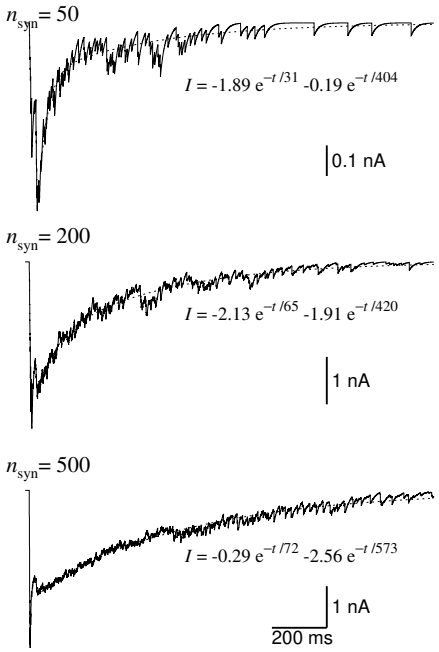


Figure 4. Reproduction of Figure 5. The precise parameters of the fitted curves are slightly different from the original, but the qualitative shape and trend in the effect of synapse number is the same.

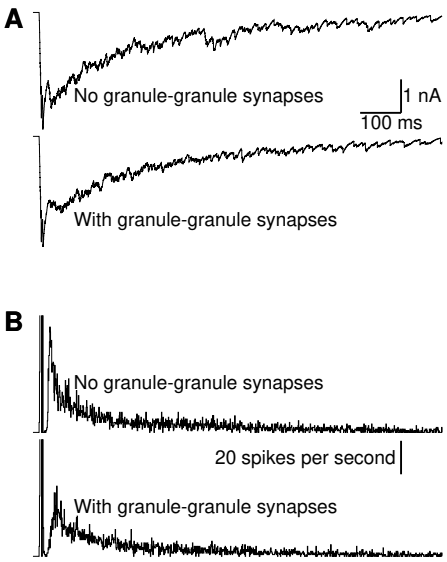


Figure 5. Reproduction of Figure 7. While there are differences in the high-frequency components of the responses, the detailed forms of the responses are very similar to the originals.

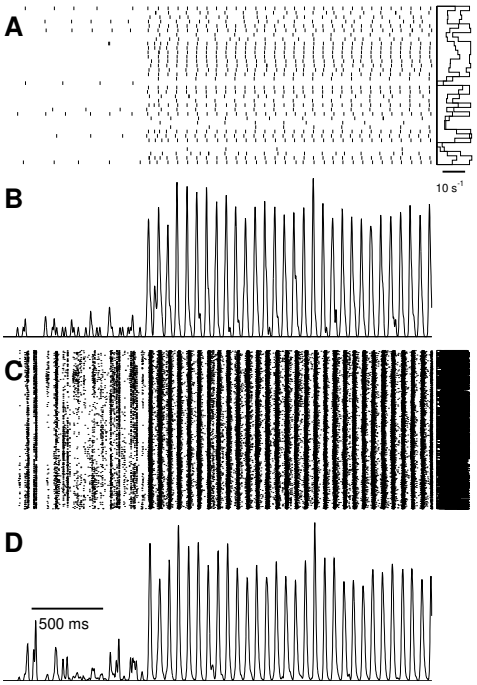


Figure 6. Reproduction of Figure 8. The precise number and timing of spikes in individual neurons are different, due to the random element in connecting the networks, but the spatial and temporal profiles of the population activity are essentially the same.

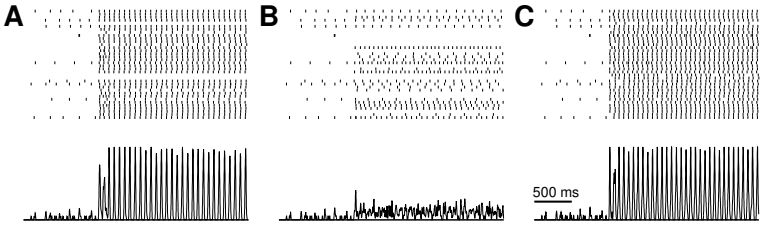


Figure 7. Reproduction of Figure 9.

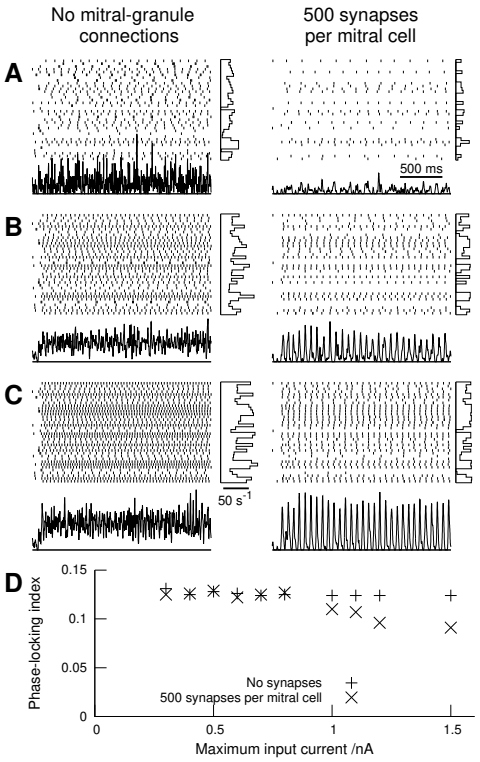


Figure 8. Reproduction of Figure 10. Small differences in the phase-locking index, especially with low input currents, are presumably due to random differences in network structure.

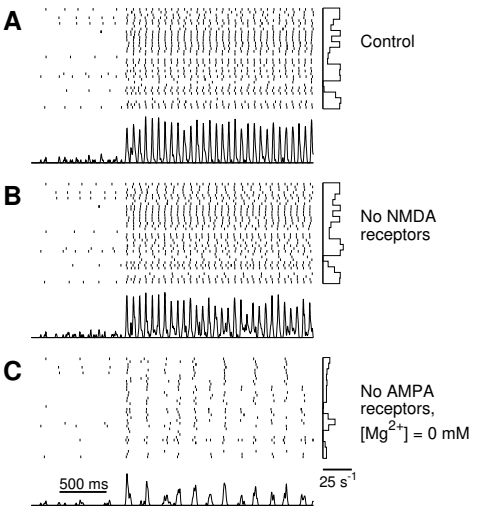


Figure 9. Reproduction of Figure 11. The effect of removing AMPA or NMDA receptors is reproduced.

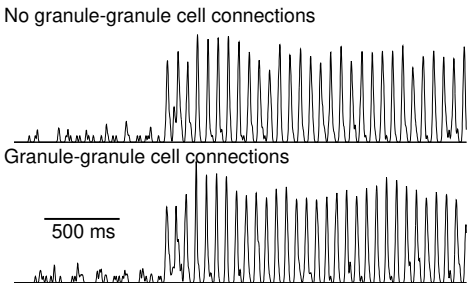


Figure 10. Reproduction of Figure 12. The results are qualitatively the same as in the original, with small differences in the timinig of individual population activity peaks.

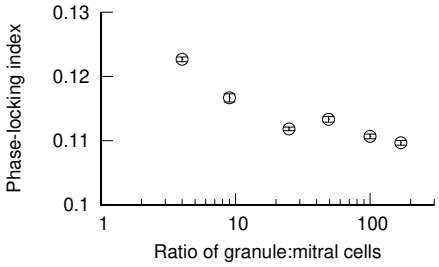


Figure 11. Reproduction of Figure 13. This is perhaps the only significant failure to reproduce. The 'U'-shaped response in the original figure is missing, and the variability of the responses is lower. Lack of time precluded a more careful examination of the sources of these differences.

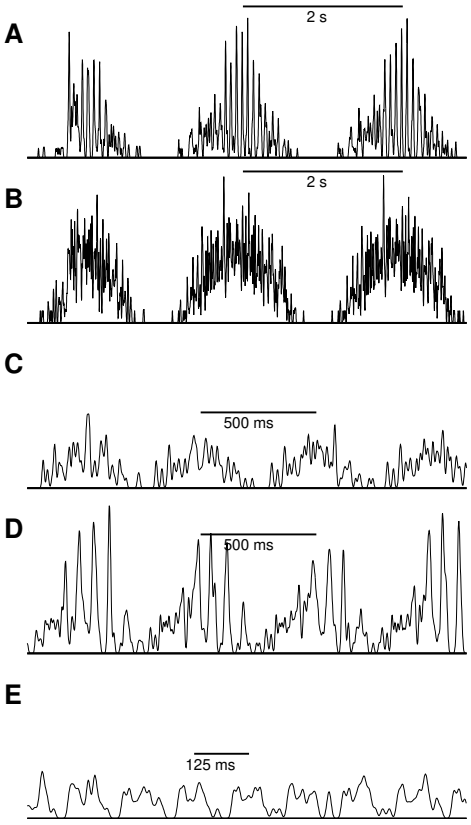


Figure 12. Reproduction of Figure 14. The qualitative features of the response are reproduced, despite small differences in high-frequency components.

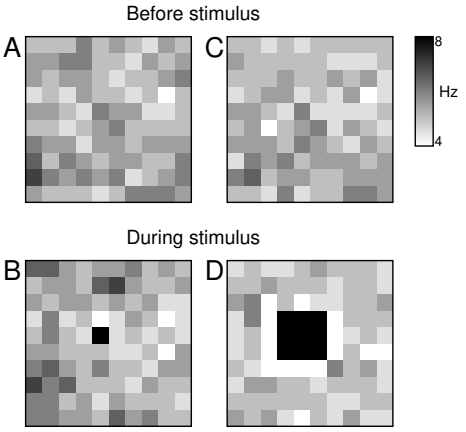


Figure 13. Reproduction of Figure 15. Some of the code used for the simulations in the original figure could not be found. The level of background input therefore had to be set by trial-and-error. There are clearly differences in the overall activity levels, but the scientific conclusion of the original, that the network structure induces functional lateral inhibition, is unchanged in the reproduction.

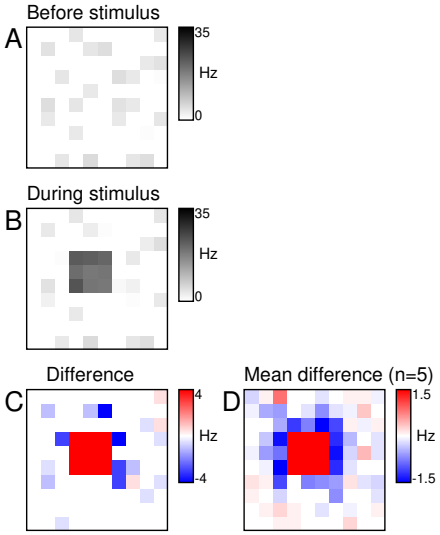


Figure 14. Reproduction of Figure 16. The same comments apply as in the caption to the previous figure.

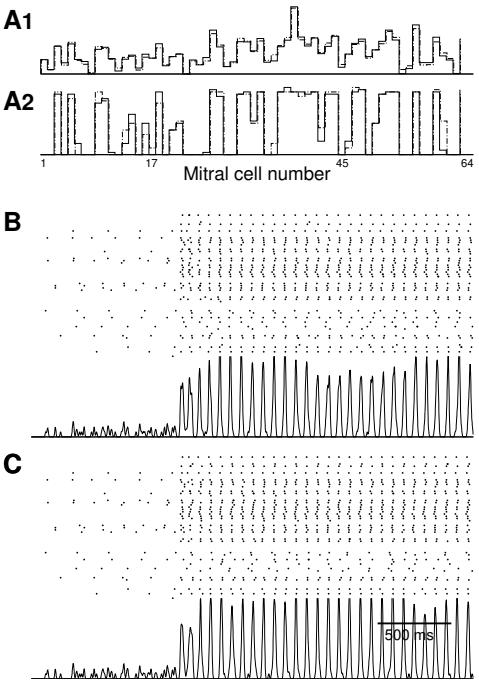


Figure 15. Reproduction of Figure 17. The assignment of odorant features ('olfactory receptors') to olfactory bulb glomeruli (and hence to mitral cells) is random in the model, and hence the input pattern is different in the reproduction compared to the original. However, the scientific conclusion of the original, that a small subset of neurons/glomeruli amplify difference between two similar odorants, is still supported by the reproduction.

tage of multi-threading in the newer versions) and Gnuplot scripts required minimal modification.

References

1. A. P. Davison, J. Feng, and D. Brown. "Dendrodendritic inhibition and simulated odor responses in a detailed olfactory bulb network model." In: **Journal of Neurophysiology** 90 (Sept. 2003), pp. 1921–1935.
2. Y. Yu, T. S. McTavish, M. L. Hines, G. M. Shepherd, C. Valenti, and M. Migliore. "Sparse Distributed Representation of Odors in a Large-scale Olfactory Bulb Circuit." In: **PLOS Computational Biology** 9.3 (Mar. 2013), pp. 1–20.
3. W. Rall. "Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic input." In: **Journal of Neurophysiology** 30.5 (Sept. 1967), pp. 1138–1168.
4. U. S. Bhalla and J. M. Bower. "Exploring parameter space in detailed single cell models: simulations of the mitral and granule cells of the olfactory bulb." In: **Journal of Neurophysiology** 69 (1993), pp. 1948–1965.
5. J. M. Bower and D. Beeman. **The Book of GENESIS: Exploring realistic neural models with the GEneral NEural SImulation System**. 2nd ed. New York: TELOS, Springer-Verlag, 1997.
6. M. L. Hines and N. T. Carnevale. "The NEURON simulation environment." In: **Neural Computation** 9.6 (1997), pp. 1179–1209.
7. A. P. Davison, J. Feng, and D. Brown. "A reduced compartmental model of the mitral cell for use in network models of the olfactory bulb." In: **Brain Research Bulletin** 51.5 (2000), pp. 393–399.
8. M. L. Hines and N. T. Carnevale. "Expanding NEURON's repertoire of mechanisms with NMODL." In: **Neural Computation** 12.5 (May 2000), pp. 995–1007.
9. M. L. Hines and N. T. Carnevale. **The NEURON Book**. Cambridge, UK: Cambridge University Press, 2006.
10. B. Ermentrout. **Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students**. Philadelphia, USA: SIAM, 2002.
11. M. L. Hines, T. Morse, M. Migliore, N. T. Carnevale, and G. M. Shepherd. "ModelDB: A Database to Support Computational Neuroscience." In: **Journal of Computuational Neuroscience** 17.1 (2004), pp. 7–11.