

Poster presentation

Open Access

PyNN: towards a universal neural simulator API in Python

Andrew Davison*¹, Pierre Yger¹, Jens Kremkow^{2,3}, Laurent Perrinet² and Eilif Muller⁴

Address: ¹UNIC, CNRS, Gif-sur-Yvette, France, ²INCM, CNRS, Marseille, France, ³Neurobiology and Biophysics, Albert-Ludwigs-University Freiburg, Freiburg, Germany and ⁴Kirchhoff Institute for Physics, University of Heidelberg, Heidelberg, Germany

Email: Andrew Davison* - andrew.davison@unic.cnrs-gif.fr

* Corresponding author

from Sixteenth Annual Computational Neuroscience Meeting: CNS*2007
Toronto, Canada. 7–12 July 2007

Published: 6 July 2007

BMC Neuroscience 2007, **8**(Suppl 2):P2 doi:10.1186/1471-2202-8-S2-P2

© 2007 Davison et al; licensee BioMed Central Ltd.

Trends in programming language development and adoption point to Python as the high-level systems integration language of choice. Python leverages a vast developer-base external to the neuroscience community, and promises leaps in simulation complexity and maintainability to any neural simulator that adopts it. PyNN <http://pynn.gforge.inria.fr/> strives to provide a uniform application programming interface (API) across neural simulators. Presently NEURON and NEST are supported, and support for other simulators and neuromorphic VLSI hardware is under development.

With PyNN it is possible to write a simulation script once and run it without modification on any supported simulator. It is also possible to write a script that uses capabilities specific to a single simulator. While this sacrifices simulator-independence, it adds flexibility, and can be a useful step in porting models between simulators. The design goals of PyNN include allowing access to low-level details of a simulation where necessary, while providing the capability to model at a high level of abstraction, with concomitant gains in development speed and simulation maintainability.

Another of our aims with PyNN is to increase the productivity of neuroscience modeling, by making it faster to develop models *de novo*, by promoting code sharing and reuse across simulator communities, and by making it much easier to debug, test and validate simulations by running them on more than one simulator. Modelers

would then become free to devote more software development effort to innovation, building on the simulator core with new tools such as network topology databases, stimulus programming, analysis and visualization tools, and simulation accounting. The resulting, community-developed 'meta-simulator' system would then represent a powerful tool for overcoming the so-called *complexity bottleneck* that is presently a major roadblock for neural modeling.

Acknowledgements

This work is supported by the European Union through the FACETS project (contract number FP6-2004-IST-FETPI-15879).