

From Bytes to Insights with Modelling as a Service

A new paradigm for computational modelling illustrated with PyRhO

Benjamin D. Evans and Konstantin Nikolic

Centre for Bio-Inspired Technology, Institute of Biomedical Engineering
Department of Electrical and Electronic Engineering, Imperial College London
London, UK

Email: {benjamin.evans, k.nikolic}@imperial.ac.uk

Abstract—Increasingly large datasets are being made publically available yet the methods and modelling skills to analyze them are lagging behind. In an effort to overcome barriers to analyzing and modelling data we propose the paradigm of *Modelling as a Service* (MaaS). As a proof-of-concept, we present a case study of the MaaS paradigm with computational tools for optogenetics based upon PyRhO. We demonstrate the benefits it confers in terms of enhanced scope for collaboration, reproducibility and ease of use, especially for scientists with a limited computational background, and discuss directions for future growth. Eventually we aim to grow this project in scope to encompass other modelling and analysis tools, and migrate to JupyterHub for persistent individual user accounts and storage. In the meantime, we hope that this approach will serve to demonstrate how MaaS can substantially increase the appeal and accessibility of modelling.

Keywords—*modelling as a service; cloud computing; portal; Jupyter; notebook; computational neuroscience; optogenetics; PyRhO*

I. INTRODUCTION

Society is accumulating data at an unprecedented rate. For example, experiments with CERN’s Large Hadron Collider alone typically generate 30 PBytes of data annually [1] with an estimated world-wide data accumulation rate in the order of tens of ZBytes/year by 2020 [2]. One particularly exciting growth area for ‘big data’ is in neuroscience. In particular, to map and characterize the brain, neuroscientists have embarked upon large-scale initiatives such as the Human Brain Project¹, the Allen Institute² and the Japan Brain/MINDS Project³, which deposit their findings into publicly available databases. However, recent studies suggest that paucity of data is not the main barrier to understanding the brain [3].

Gaining insight from these data requires modelling, analysis and sharing of results in a similarly transparent and reproducible way. However, typically complex installation requirements and a lack of programming experience may hinder the adoption of such modelling practices to the detriment of the field. The work presented here aims to overcome these barriers to adoption through the paradigm of “Modelling as a Service” (MaaS), whereby the necessary tools are ready and waiting as a webpage in the cloud.

Recently, the IPython/Jupyter project [4] has evolved into a browser-based notebook interface where the inputs and outputs of the modelling process are all embedded into the same webpage (along with a potentially wide variety of other rich web content). This auto-documenting approach to simulation and analysis has been identified as an especially promising medium for promoting sharing and reproducibility in computational neuroscience [5]. As such, these technologies are natural choices for creating a MaaS platform.

This approach was further enhanced by the development of *JupyterHub* and *tmpanb* [6] which handle authentication for multiple simultaneous users, spawning separate (temporary) notebooks for each in secure Docker sandbox environments. These systems are open-source and may be easily adapted to a particular scientific application by substituting in a tailored base notebook image with the required packages. As such the development of these technologies has realized the possibility of making all the tools for MaaS available.

We demonstrate the MaaS paradigm with Prometheus: a web-based portal for optogenetics, (including a graphical user interface), configured with all the necessary Python dependencies preinstalled, such that they may be used on any computer with nothing more than a web browser and an Internet connection. The work presented greatly simplifies the use of modelling software, providing a readily accessible alternative to local installations for quick trials without the need to download, install or configure any new software. We illustrate the benefits of such an approach and discuss the limitations, with suggestions for future enhancements.

II. CASE STUDY: COMPUTATIONAL OPTOGENETICS

As a proof-of-concept of the MaaS paradigm for more accessible modelling, we developed an online portal for optogenetics dubbed *Prometheus* (<http://try.projectpyrho.org>), based on *PyRhO*, our computational tools for fitting and simulating opsins [7]. Optogenetics is a field of neuroscience whereby neurons are rendered sensitive to light through transfection with genes coding for light-sensitive proteins (originally known as rhodopsins). The combination of genetic targeting and light-mediated actuation makes this technique both highly spatially and temporally precise, allowing the role of individual classes of neurons to be reversibly controlled *in vivo*. This has proven to be an extremely powerful tool, with widespread applications throughout the biological sciences and was declared the “Method of the Year” by Nature Methods [8].

¹ www.humanbrainproject.eu

² www.alleninstitute.org

³ <http://brainminds.jp/en/>

To make the technology even more effective, new variants of these light-sensitive proteins are continually being engineered, which previously had to be painstakingly characterized through many careful experiments. PyRhO allows for a significant reduction in the number of such experiments by parameterizing kinetic Markov models of the (rhod)opsin’s dynamics from a limited set of data. The models may then be used to test the suitability of the opsins for a particular application *in silico*, thereby avoiding more time-consuming experiments and accelerating the progress of the field. In the process of characterization, insights may also be gained regarding the effects of mutations at a particular locus.

At the core of PyRhO are three functional kinetics Markov models describing the dynamics of (rhod)opsins. These opsin models have been integrated with several popular simulation environments including NEURON [9, 10] and Brian2 [11, 12], allowing simulations to be run on multiple scales (from channel to network). An array of customizable stimulation protocols is also included, allowing the models to be characterized with a range of stimuli. Notably, PyRhO also includes a Graphical User Interface (GUI) based on IPython widgets, providing an alternative to programming models and thereby further enhancing the usability for neuroscientists with a limited computational background.

III. METHODS

Given the surge in popularity of Python as a programming language (particularly in neuroscience) mainly due to its expressiveness, readability, and wealth of open-source modules [13], this was a natural foundation on which to build PyRhO and Prometheus. The Jupyter notebook is also particularly well suited to creating more widely useful portals, (particularly for data science and machine learning), due to its versatility in working with approximately 70 other computational kernels, including for example, R, Julia and Go.

Several options are available for creating a scientific notebook-based portal within the Jupyter project, namely *JupyterHub* and *tmpnb*. The main difference between these choices concern authentication of users and persistence of notebooks. While JupyterHub is designed to only allow access to registered users and to save their work, *tmpnb* can be configured to serve a template notebook for any user to experiment with, which is automatically deleted after a period of inactivity, freeing resources for the next user.

While JupyterHub is the more fully-featured, substantial solution, it requires more resources in terms of storage space and entails more of an administrative overhead in terms of user management. Baring authentication and data persistence, without further loss of generality, the capabilities of notebooks for Modelling as a Service may be more easily demonstrated with *tmpnb*, which has the added convenience of automatically deleting inactive notebooks to keep resource usage to a minimum.

In both cases, Docker containers are used to run all processes in isolated, reproducible sandboxes. This software has become extremely popular as a virtualization and application packaging technology since images are lightweight and their creation is scriptable, allowing them to be placed

under version control. This ensures that the computational environment is consistent, irrespective of the underlying hardware. They may also be easily built upon other images to leverage previous work in a granular way. This makes Docker a natural choice to aid in the reproducibility of models.

In the present work, the HTTP Proxy listens on the public interface and forwards requests to the *tmpnb* orchestrator server which listens on a private REST (representational state transfer) API interface. The orchestrator assigns a notebook server in a Docker container from a pool (or spawns a new one) and configures the proxy to redirect the client there. After a timeout period, inactive notebook servers are destroyed, with fresh ones being created as required. For a schematic overview, see Fig. 1. The notebook Docker container is based on a customized image which contains PyRhO and all of its dependencies which may be easily updated and extended as required.

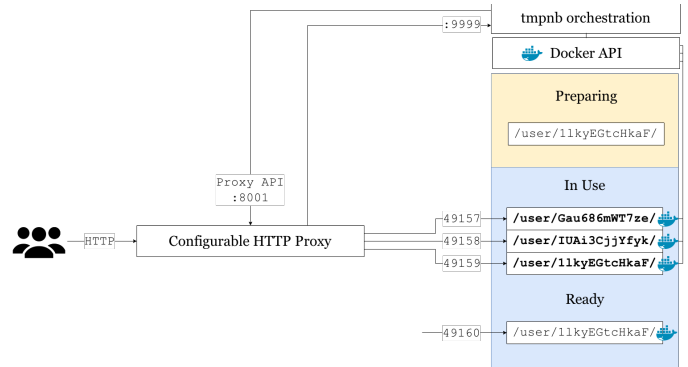


Fig. 1. The *tmpnb* architecture schematic (reproduced from <https://github.com/jupyter/tmpnb>) showing the proxy server, orchestration server and pool notebook servers in Docker containers.

The system presented here is built upon a Linux server using an entirely open-source software stack, resulting in a robust, responsive and easily replicable system. The complete stack is illustrated in Fig. 2. The code, scripts and configuration files used to build the Docker image and portal (including PyRhO) are available at the project’s GitHub repository: <https://github.com/ProjectPyRhO>, enabling others to rapidly adapt and deploy the system for both research and teaching.

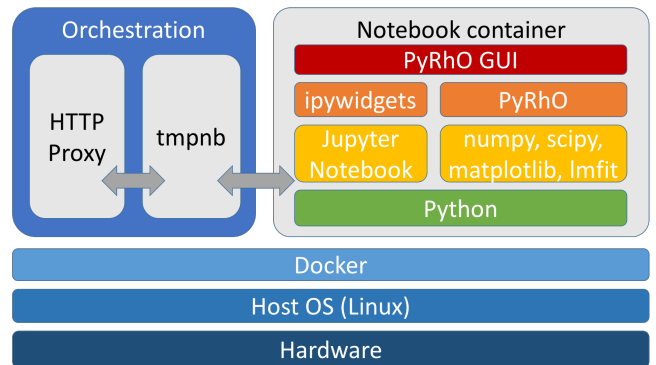


Fig. 2. Prometheus Software Stack. The core system is shown in blue (from Hardware to the Orchestration containers) with Docker containers shown in grey. The notebook container in this work uses Python and the PyRhO module but may be easily adapted to alternative languages and modules.

IV. RESULTS

As a specific illustration of general concepts, we present the modes of usage which Prometheus offers and the advantages they represent. Like other online portals, this approach eliminates problems associated with local installations, such as lack of administrative permissions, version incompatibilities and unsupported operating systems to name a few. Unlike other portals, the notebook interface is lightweight, interactive and provides the opportunity to be controlled entirely through an intuitive GUI composed of familiar widgets as shown in Fig. 3. We illustrate the advantages of such a system in terms of its interactivity, convenience and scope for easily sharing and reproducing models and analyses.

A. System Identification

The functional kinetic Markov models which describe the state-transitions of the opsins are parameterized according to experimental data. If a new opsin's behavior is recorded, the data may be fit to the models using user-supplied or default initial parameters. Prior information may be built into the fitting process by selectively fixing parameters, setting bounds and even algebraic constraints, with the optimization algorithms then finding the remaining free parameters which best describe those data. This gives a complete model which may then be seamlessly used as the basis for simulations within the same framework.

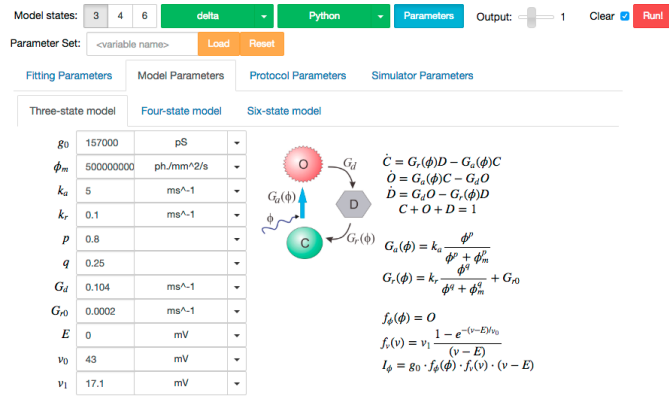


Fig. 3. The PyRHO GUI running in a browser-based notebook. The run bar (for controlling simulations) can be seen at the top and the model parameters can be set below with familiar widgets (including buttons, sliders and drop-down menus).

B. In Silico Experiments

Once a model has been fit (or chosen from a set of pre-fit options), this can be embedded into a more complex model for computational simulations at multiple scales. It is especially important to make this step as close to the first as possible so that new ideas can be quickly tested in order to reveal shortcomings of computational models, suggest refinements or prompt questions to be answered with further data collection. This closes the loop between data and model bringing experimental and theoretical work into harmony.

An example of the results is shown in Fig. 4. Here we also demonstrate how the internal states of the models may be analyzed, characterizing the internal state transition processes which give rise to the characteristic peak-plateau opsin

photocurrent. This allows scientists to measure internal states to test ideas in simulations which may otherwise be very difficult or impossible to explore experimentally. Gaining direct access to the internal states of such models serves as an example of how hypotheses may be tested and insights may be gained from the modelling process.

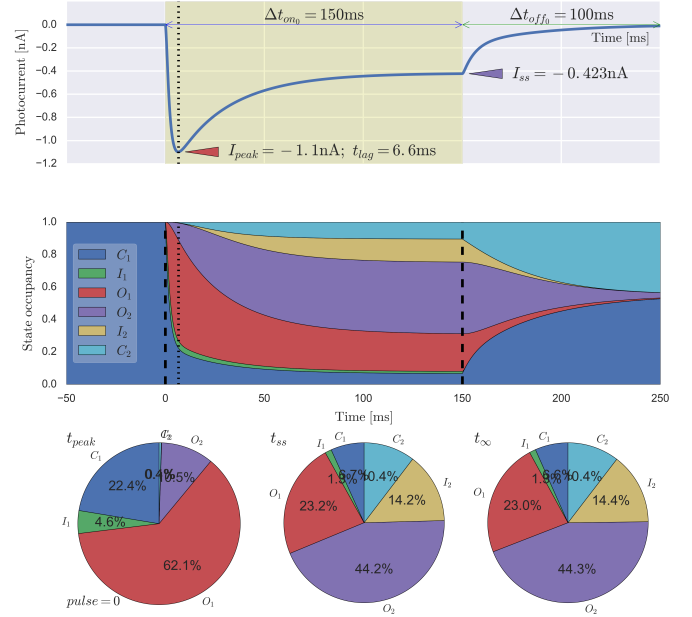


Fig. 4. Evolution of photocurrent (top row) and state variables through time (middle row) with the proportions at the peak, pulse end and analytic steady-state conditions (bottom row). The opsins are initially in their dark-adapted state ($C_1=1$, dark blue) then once illuminated (left dashed line), the models transition to their open states (predominantly O_1 , red, also O_2 , purple), via intermediate states (I_1 , green and I_2 , yellow) causing the photocurrent (top) to peak (dotted line, bottom-left pie chart; t_{peak}) then settle to a steady-state (bottom-centre) as an equilibrium is reached for transitions in and out of the open states. The centre pie chart (t_{ss}) shows the simulated state occupancy proportions at light-off and the right pie chart (t_{∞}) shows the analytic occupancies as t tends to infinity. When the illumination is turned off (right dashed line) the opsins transition to their closed states (C_1 , dark blue and C_2 , light blue), and will eventually return to their ground state (C_1).

C. Multi-platform Integration

An important advantage of this approach to modelling, is the freedom to use specific tools, versions and even bespoke modifications in order to make them communicate together without interfering from pre-existing installations. For example, in order to integrate NEURON with code written in Python 3, it's source code had to be modified and compiled as the pre-compiled binaries would only work with Python 2. This may have created conflicts with local installations if users already had the simulator on their personal computers. In contrast, Docker containers are designed to encapsulate computational environments, allowing precise control of the installed versions and configuration settings of each dependency.

An example of integrating the Brian2 spiking neural network simulator is shown in Fig. 5 where a chirp stimulus is presented to the transfected (leaky integrate-and-fire) neurons of the input layer and the resulting waves of spikes can be seen to propagate through successive layers.

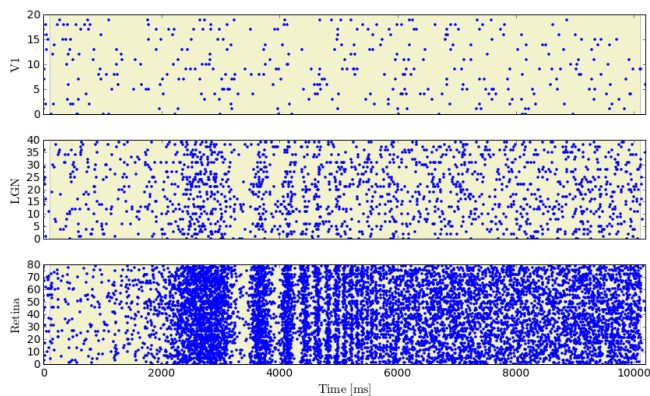


Fig. 5. A three-layer network simulation with Brian2 modelling the first three layers of the visual system: the retina, lateral geniculate nucleus (LGN) and primary visual cortex (V1). The input layer (retina) is transfected with Chr2 (modelled with the three-state model) and stimulated with a chirp.

D. Reproducibility and Sharing

Since the system is built on Docker containers, the exact computational environment can be easily replicated while the notebook format automatically logs the inputs along with the generated outputs (typically as embedded figures). The commands and results may also be supplemented with notes, equations, web-links, pictures, videos and various other rich web content to annotate and describe the scientific contents. The notebooks may then be exported in a variety of formats including as md, pdf or ipynb (native) for sharing, or linked to on *nbviewer* (<https://nbviewer.jupyter.org/>) as static html.

V. DISCUSSION

We have presented Prometheus as a realization of a more general paradigm in computational modelling. The example given demonstrates many of the advantages possible with such a cloud-based scientific notebook including:

- No local installation or configuration necessary
- Standard permissions are sufficient
- Requires only a browser and Internet connection
- Greater reproducibility of work due to version control
- Enhanced capabilities for sharing
- Intuitive GUI with accompanying rich web content
- Simplified usage with low barrier to adoption

While the paradigm presented here may not fully replace local installations, which still offer a number of advantages (e.g. lower latency, continuity, data security), it does provide a readily accessible alternative which can be quickly tried without the need to download, install or configure any new software, substantially increasing its appeal for scientists with limited computational experience. This may also be useful for quick “on-the-go” prototyping of models inspired by a conference or when otherwise away from the office.

There are several challenges in creating such a system. Security may be a concern for some, since algorithms and data are placed on an Internet-facing server. However, Docker contains are secure by design and the same concerns apply to any workstation with an Internet connection. Using JupyterHub would also provide an additional layer of security through

authentication, as discussed. Another challenge is posed by the expensive hardware requirements of a multi-user server, which exceed those of standard desktops. However, as the user-base grows the efficiencies of scale and virtualization could ultimately render this approach much more cost-effective.

In the future, this prototype would benefit by moving to the more substantial JupyterHub for management of user accounts and persistence of data. For example, it could augment a system such as the Neuroscience Gateway (NSG) [14] providing a graphical, interactive computing facility on very powerful hardware. While this would require further ongoing funding, in the meantime, the example given here may be used as a template and adapted for individual laboratories to setup and tailor to their needs. In this way, we hope that MaaS will help to promote a tighter integration between experimental and modelling work, accelerating progress by transforming bytes into insights.

ACKNOWLEDGMENT

The work was kindly supported by the UK BBSRC grant: BB/L018268/1, the BBSRC Impact Acceleration Award and the UK EPSRC grant: EP/N002474/1. We would also like to thank the developers of Project Jupyter (<https://jupyter.org/>) for building an excellent foundation upon which Prometheus rests.

REFERENCES

- [1] CERN, "Computing," 2016. [Online]. Available: <http://home.cern/about/computing>. [Accessed 21 June 2016].
- [2] CSC, "Big Data Universe Beginning To Explode," 2012. [Online]. Available: http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode. [Accessed 21 June 2016].
- [3] E. Jonas and K. Kording, "Could a neuroscientist understand a microprocessor?," *bioRxiv*, vol. 055624, 2016.
- [4] F. Pérez and B. E. Granger, "IPython: A System for Interactive Scientific Computing," *Computing in Science & Engineering*, vol. 9, pp. 21-29, May/June 2007.
- [5] M. Topalidou, A. Leblois, T. Boraud and N. P. Rougier, "A long journey into reproducible computational neuroscience," *Frontiers in Computational Neuroscience*, vol. 9, no. 30, March 2015.
- [6] H. Shen, "Interactive notebooks: Sharing the code," *Nature*, vol. 515, pp. 151-152, 6 November 2014.
- [7] B. D. Evans, S. Jarvis, S. R. Schultz and K. Nikolic, "PyRhO: A Multiscale Optogenetics Simulation Platform," *Frontiers in Neuroinformatics*, vol. 10, no. 8, March 2016.
- [8] Editorial, "Method of the Year 2010," *Nature Methods*, vol. 8, no. 1, January 2011.
- [9] M. Hines, A. P. Davison and E. Muller, "NEURON and Python," *Frontiers in Neuroinformatics*, vol. 3, no. 1, January 2009.
- [10] N. T. Carnevale and M. L. Hines, *The NEURON Book*, Cambridge: Cambridge University Press, 2006.
- [11] D. F. M. Goodman and R. Brette, "Brian: a simulator for spiking neural networks in Python," *Frontiers in Neuroinformatics*, vol. 2, no. 5, November 2008.
- [12] M. Stimberg, D. F. M. Goodman, V. Benichoux and R. Brette, "Equation-oriented specification of neural models for simulations," *Frontiers in Neuroinformatics*, vol. 8, no. 6, February 2014.
- [13] E. Muller, J. A. Bedner, M. Diesmann, M.-O. Gewaltig, M. Hines and A. P. Davison, "Python in Neuroscience," *Frontiers in Neuroinformatics*, vol. 9, no. 11, April 2015.
- [14] S. Sivagnanam, A. Majumdar, K. Yoshimoto, V. Astakhov, A. Bandrowski, M. A. Martone and N. T. Carnevale, "Introducing The Neuroscience Gateway," in *International Workshop on Science Gateways*, Zurich, 2013.