INCF SBP Letter in support of NetPyNE

Authors: Joe Graham and Salvador Dura-Bernal
Emails: joe.w.graham@gmail.com, Salvador.Dura-Bernal@downstate.edu

SBP author: Salvador Dura-Bernal and members of the Dura-Bernal Lab and Neurosim Lab
Author email: Salvador.Dura-Bernal@downstate.edu
Organization: State University of New York (SUNY) Downstate Health Sciences University
Website: http://dura-bernal.org/ and http://neurosimlab.org

SBP name: NetPyNE (Networks using Python and NEURON)
SBP GitHub: https://github.com/Neurosim-lab/netpyne
SBP website: http://netpyne.org/

Brief description: NetPyNE is an open-source Python package to facilitate the development, parallel simulation, analysis, and optimization of multiscale biophysical neuronal networks using the NEURON simulator. NetPyNE includes a standardized declarative language for high-level specifications of brain circuit models, which is the focus of this proposal.

Stewards: Major decisions about NetPyNE are made by the steering committee, guided by the Project roadmap and the Code of conduct. The committee includes members from a diverse range of institutions, positions and backgrounds.

The current steering committee consists of the following members (in alphabetical order):

- Salvador Dura-Bernal (Assistant Professor, State University of New York Downstate; Research Scientist, Nathan Kline Institute for Psychiatric Research)
- Padraig Gleeson (Principal Research Fellow, University College London)
- Joe W. Graham (Research Scientist, State University of New York Downstate)
- Erica Y. Griffith (Graduate Student, State University of New York Downstate)
- Michael Hines (Senior Research Scientist, Yale University)
- Cliff C. Kerr (Senior Research Scientist, Institute for Disease Modeling)
- William W. Lytton (Distinguished Professor, State University of New York Downstate; Kings County Hospital)
- Robert A. McDougal (Assistant Professor, Yale University)
- Samuel A. Neymotin (Research Scientist, Nathan Kline Institute for Psychiatric Research)
- Benjamin A. Suter (Postdoctoral Fellow, Institute of Science and Technology Austria)
- Subhashini Sivagnanam (Principal Computational and Data Science Research Specialist, San Diego Supercomputing Center)

Membership in the steering committee is a personal membership. Affiliations are listed for identification purposes only; steering committee members do not represent their employers or academic institutions.

Relevant publication: Dura-Bernal S, Suter BA, Gleeson P, Cantarelli M, Quintana A, Rodriguez F, Kedziora DJ, Chadderdon GL, Kerr CC (2019) NetPyNE, a tool for data-driven multiscale modeling of brain circuits, eLife 2019;8:e44494. https://doi.org/10.7554/eLife.44494

Nomination information
NetPyNE was nominated for consideration by Salvador Dura-Bernal on March 22, 2022.

About NetPyNE

NetPyNE is an open-source Python package to facilitate the development, parallel simulation, analysis, and optimization of biophysical neuronal networks using the NEURON simulator. The tool includes the following components:

- Standardized declarative language for high-level specifications: enables the user to define neuronal circuit models across scales in a compact manner (e.g. using connectivity rules instead of a list of all connections); JSON-like format.
- Standardized data structure for network instance: generated by NetPyNE from the high-level specifications; contains the NEURON objects required to run the simulation; enables the user to explore and modify specific network instance components after creation; be imported and exported to NeuroML language.
- Standardized data structure for simulation output: generated by NetPyNE from the NEURON simulation output; enables the user to explore and modify the simulation outputs.
- Application programming interface (API): Python functions that enable the user to create, manipulate, load and save the high-level specifications, network instance and simulation output; run NEURON parallel simulations and automated parameter optimizations/explorations; export/import to other formats; and run analysis and plotting functions.
- Graphical user interface (GUI): HTML5 and Javascript tool that enables the user to perform the same functionalities as the NetPyNE API described above, but through a web-based graphical interface.

This proposal is focused on the *standardized declarative language for high-level specifications* and the *standardized data structure for network instance*.

NetPyNE conforms to the main criteria outlined by INCF for consideration for endorsement. It is open-source and designed around the FAIR principles of Findability, Accessibility, Interoperability, and Reusability. NetPyNE is well documented with good test coverage and is governed by a volunteer steering committee led by documented goals and standards. There are multiple forums in which NetPyNE users and developers interact. A number of tutorials are available including interactive Jupyter notebooks and YouTube videos. There is a large and growing user community with dozens of ongoing projects utilizing NetPyNE. NetPyNE is currently supported by a five-year grant from the National Institutes of Health (NIH), and we will apply to extend that grant. While there are other INCF SBPs that cover portions of the capabilities, NetPyNE is unique in multiple modeling scales (from molecules to brains) and its breadth of function (import/export, network specification, parallel simulation, variety of analyses, and optimization of parameters).

Open criteria

1. **Is the SBP covered under an open license so that it is free to implement and reuse by all interested parties (including commercial)?**
   Yes.

2. **What license is used?**
   NetPyNE uses the MIT license (https://opensource.org/licenses/MIT).
3. **Does the SBP follow open development practices?**
   Yes, the specification and all code are developed and released on GitHub.
4. **Where and how are the code/documents managed?**
   All code and documentation is on GitHub (https://github.com/Neurosim-lab/netpyne). Active development is done in the *development* branch (https://github.com/Neurosim-lab/netpyne/tree/development). When changes have been tested and documented they are incorporated into the *master* branch. Stable releases are tagged on GitHub and added to PyPI approximately monthly (https://pypi.org/project/netpyne/). Issues are tracked on GitHub (https://github.com/Neurosim-lab/netpyne/issues) where contributors can make pull requests (https://github.com/Neurosim-lab/netpyne/pulls). The documentation is stored on GitHub (https://github.com/Neurosim-lab/netpyne/tree/development/doc) and built using Sphinx (https://www.sphinx-doc.org/).
5. **Any additional comments on the openness of the SBP?**
   NetPyNE actively encourages contributions from users, from bug reports to feature requests to contributions to the code and pull requests. Please see the NetPyNE Contributor's Guide for details: https://github.com/Neurosim-lab/netpyne/blob/development/CONTRIBUTING.md.

FAIR criteria

Considers the SBP from the point of view of some (not all) of the FAIR criteria (Wilkinson et al. 2016). Is the SBP itself FAIR? Does it result in the production of FAIR research objects? Note that many of these may not apply. If so, leave blank or mark N/A.

1. **SBP uses/permits persistent identifiers where appropriate**
   NetPyNE's declarative language includes a field (*netParams.version)* to uniquely identify the version of the model being developed. Additionally, every NetPyNE simulation includes the specific version of NetPyNE used and a user-generated label (*cfg.simLabel*) and description which may be used to uniquely identify the simulation run. NetPyNE network instances can also be output to NeuroML format, which permits persistent identifiers within files expressed in the language.
2. **SBP allows addition of rich metadata to research objects**
   N/A (see 10)
3. **SBP uses/permits addition of appropriate PIDs to metadata**
   N/A (see 10)
4. **The protocol allows for an authentication and authorization when required**
   NetPyNE is now integrated into Open Source Brain (Open Source Brain) and the EBRAINS platform, which provide cloud-based workspaces that allow for authentication and authorization of users.
5. **SBP uses or allows the use of vocabularies that follow the FAIR principles**
   N/A (see 10)
6. **SBP includes/allows qualified links to other identifiers**
   N/A (see 10)
7. **Does the standard interoperate with other relevant standards in the same domain?**

Yes, NetPyNE can import from and export to the NeuroML standard and the SONATA standard. NetPyNE is also interfaced with several other tools and platforms in the field, including Open Source Brain (OSB), EBRAINS, The Neuroscience Gateway (NSG), The Virtual Brain (TVB), the Human Neocortical Neurosolver (HNN), SciUnit/SciDash, LFPy and coreNEURON. There is also work in progress to interoperate with NWB format for input and output data.

8. **Does the SBP provide citation metadata so its use can be documented and tracked?**
   NetPyNE has an RRID (Research Resource IDentifier https://www.rrids.org/): RRID:SCR_014758
9. **Does the SBP have a clear versioning scheme and appropriate documentation?**
   NetPyNE uses standard semantic versioning (current stable release version: 1.0.1; see list of releases: https://github.com/suny-downstate-medical-center/netpyne/releases), has extensive documentation on general usage (http://netpyne.org/reference.html) and of the API (http://netpyne.org/package_index.html), and has a number of tutorials available online. (http://netpyne.org/tutorial.html)
10. **Any additional comments on aspects of FAIR?**
    For questions 2, 3, 4 and 6, we note that NetPyNE's declarative language includes a field (*netParams.description*) that allows to store any arbitrary text, including metadata, PIDs, vocabularies and qualified links.This information could also be potentially added in the declarative language at the level of individual cell property, connectivity and stimulation rules, which allow to include arbitrary fields.
    NetPyNE was designed to facilitate model sharing, reusability, interoperability and reproducibility by providing a high-level declarative standardized language for multiscale neuronal network models. This language was designed to address many of the intricacies of experimental data, such as complex connectivity rules, the distribution of synapses across fully detailed dendrites, and time-varying stimulation. Contrasting with the obscurity of raw-code descriptions used in many existing NEURON models (e.g. in ModelDB), NetPyNE's standardized language provides transparent and manageable specifications that are easier to find, access, interoperate and reuse. Model specifications are then translated into the necessary NEURON components via built-in well-tested algorithms. This approach cleanly separates high-level model specifications from the underlying technical implementation. This unique feature of NetPyNE partly motivated the ongoing development of NeuroMLLite, an extension of NeuroML to incorporate similar high-level network model descriptions. Additionally, we are implementing automated validation of NetPyNE's declarative language (https://github.com/suny-downstate-medical-center/netpyne/blob/development/netpyne/sim/validator.py) to ensure appropriate format and syntax, and to indicate issues and make suggestions to the user, further ensuring that the specifications follow FAIR principles.

Design, Testing and Implementation

1. **What is the technical expertise level required to implement this? Even if it is quite difficult, should it be implemented anyway?**
   The implementation requires significant technical expertise since it involves in-depth knowledge of the NEURON simulation engine and how to implement neural components at multiple scales (from RxD species, to ionic channels and synapses, to detailed neurons to complex circuit connectivity). The NetPyNE package already provides an implementation, which has been widely used and has been shown to be useful to the neuroscience community.
2. **Does the SBP provide an architectural concept to understand its implementation and relationships to external entities?**

Yes, both the NetPyNE architectural implementation and its relationships with external entities are extensively described in the NetPyNE publication (https://elifesciences.org/articles/44494) and the package reference (http://netpyne.org/reference.html), including diagrams of the overall architecture, high-level specifications and instantiated network.

3. **Does the SBP have a reference implementation?**
   Yes, the reference implementation is provided by the NetPyNE codebase (https://github.com/Neurosim-lab/netpyne), which translates the high-level specifications into a NEURON network instance, i.e. it generates a Python structure containing the required NEURON objects from the NetPyNE standardized language. There is a JSON-based schema of the high-level specification language (https://github.com/Neurosim-lab/netpyne/blob/development/netpyne/metadata/metadata.py), and we are currently developing a more elaborate schema (https://github.com/suny-downstate-medical-center/netpyne/blob/development/netpyne/sim/validator.py) using the *Schema* python package. Both of these schemas can be used for automated validation purposes (e.g. the JSON schema is already employed by the GUI). The network instance and output data structure is also described in the documentation (http://netpyne.org/reference.html#netpyne-data-model-structure-of-instantiated-network-and-output-data).
4. **What software resources (files/scripts/libraries/tools) are available to support the SBP?**
   NetPyNE's codebase (https://github.com/Neurosim-lab/netpyne) includes a large set of subpackages (API) to operate with the NetPyNE declarative language for high-level specifications and NetPyNE's network instance and output simulation data. These range from creating the network, to running simulations, to parameter optimization, to analyzing the simulation output. This API is also available via the Open Source Brain (OSB; https://v2.opensourcebrain.org/) and EBRAINS platforms (https://ebrains-cls-interactive.github.io/online-use-cases.html#/small_circuit_insilico_experiment). Because NetPyNE can import from and export to NeuroML and SONATA, it can be used with 25+ NeuroML compliant tools/libraries/databases here: https://www.neuroml.org/tool_support. NetPyNE also has its own graphical user interface (GUI) (http://www.netpyne.org/gui) and is now integrated into OSB.
5. **Are the supporting software resources tools and implementations covered under an open source license?**
   Yes, all supporting software (e.g. NEURON, NumPy, Matplotlib) is open source.
   a. **Are the supporting software resources well documented (documentation of I/O operations, programming interfaces, user interfaces, installation)?**
      Yes, NetPyNE only uses common, well-supported, well-documented supporting software.
   b. **Were the supporting software resources validated?**
      Not by NetPyNE developers, but NetPyNE only uses common, well-supported, well-documented supporting software.
   c. **What is your assessment of the quality of the code/document?**
      NetPyNE has a code standard based largely on PEP8 and most legacy code has been properly formatted. The process is ongoing and supported by a 5-year NIH grant focused on tool robustness and following software development best practices. We are in the process of applying the *black* python package to automatically enforce PEP8 on all code (https://github.com/suny-downstate-medical-center/netpyne/tree/blackened). Docstrings stubs are in place for every code object, are in NumPy style, and are used to automatically create online documentation for the API. There is also an extensive general reference (http://netpyne.org/reference.html).

   d. **Have the supporting software resources been deployed, is there any experience or references to their use by the community?**
      All supporting software is open-source, freely available and has been deployed. These supporting software resources have been widely used by the community -- see Adoption of Use section below.
6. **Any additional comments on design, testing, and implementation?**
   NetPyNE follows continuous integration, including the use of Github Actions, unit testing and GitHub workflows to incorporate community contributions (Pull Requests, reviews, etc).

Governance

1. **Does the SBP have a clear description on who is maintaining the SBP and how decisions regarding its development are made?**
   Yes, major decisions about NetPyNE are made by the steering committee, guided by the Project roadmap and the Code of conduct. The committee includes members from a diverse range of institutions, positions and backgrounds.
2. **Is the governing model document for maintenance and updates compatible with the INCF project governing model document and the open standards principles?**
   Yes, the NetPyNE contributor's guide (https://github.com/Neurosim-lab/netpyne/blob/development/CONTRIBUTING.md) and the code of conduct (https://github.com/Neurosim-lab/netpyne/blob/development/CONTRIBUTING.md) are compatible with INCF standards.
3. **Is the SBP actively supported by the community? If so, what is the evidence?**
   Yes, there are over 39 contributors who have had improvements committed (https://github.com/suny-downstate-medical-center/netpyne/graphs/contributors). Many users have submitted bug reports and feature requests -- there are currently 170 closed and 110 open issues (https://github.com/Neurosim-lab/netpyne/issues). Many users have made pull requests -- there are currently 381 closed and 8 open pull requests (https://github.com/Neurosim-lab/netpyne/pulls).
4. **Does the SBP provide tools for community feedback and support?**
   Yes, in addition to the Github issue tracker and pull requests, NetPyNE also maintains a mailing list (https://groups.google.com/g/netpyne-mailing), a question-and-answer forum for NetPyNE (https://groups.google.com/g/netpyne-forum), and for NEURON+NetPyNE (https://www.neuron.yale.edu/phpbb/viewforum.php?f=45), and we maintain a Twitter account (https://twitter.com/_NetPyNE_).
5. **Any additional comments on governance?**
   Contributors to NetPyNE are members of the INCF SIG on network specifications: https://www.incf.org/activities/standards-and-best-practices/incf-special-interest-groups/incf-sig-on-standardised.

Adoption and Use

1. **Is there evidence of community use beyond the group that developed the SBP?**
   Yes, there is an established community of users beyond the original developers of NetPyNE. This is evidenced by the mailing list (https://groups.google.com/g/netpyne-mailing), Q&A forums (https://groups.google.com/g/netpyne-forum; https://www.neuron.yale.edu/phpbb/viewforum.php?f=45), and Github issues activity

(https://github.com/Neurosim-lab/netpyne/issues). In addition to specific projects and publications listed below, NetPyNE is actively collaborating with the Open Source Brain project, a resource for sharing and collaboratively developing computational models of neural systems (https://v2.opensourcebrain.org/), the Human Brain Project E-BRAINS infrastructure ((https://ebrains-cls-interactive.github.io/online-use-cases.html#/small_circuit_insilico_experiment) , the Human Neocortical Neurosolver (https://github.com/jonescompneurolab/hnn/tree/hnn2), the Blue Brain Project coreNEURON (https://github.com/BlueBrain/CoreNeuron), The Virtual Brain (www.thevirtualbrain.org/), and SciUnit/NeuronUnit (https://github.com/scidash/sciunit; https://github.com/lakesare/netpyneunit). NetPyNE has also been used as an educational and training tool at multiple courses and workshops: http://netpyne.org/about.html#courses.

2. **Please provide some concrete examples of use, e.g., publications where the use of the SBP is cited; databases or other projects that have adopted the SBP**
   We maintain a spreadsheet of known projects utilizing NetPyNE with 97 models and over 90 unique users across over 40 institutions worldwide (http://netpyne.org/models/). There are currently 7 publications about NetPyNE (http://netpyne.org/about.html#publications) and at least 25 publication making use or citing NetPyNE, many of which from researchers beyond the group that developed NetPyNE (http://netpyne.org/about.html#make-use-of-netpyne).
3. **Is there evidence of international use?**
   Yes, there is a wide international spread of users, contributors, and developers, as evidenced in the links provided above. There is a large user base, for example, from South America, particularly Brazil, due to several courses organized in the country (LASCON; http://sisne.org/lascon-viii/?lang=en).
4. **Any additional comments on use?**

Stability and Support

1. **Who is responsible for maintaining the SBP?**
   Maintenance of NetPyNE is the responsibility of the Dura-Bernal lab (http://dura-bernal.org/) and the Neurosim lab (http://www.neurosimlab.com/) at SUNY Downstate Health Sciences University.
2. **How is it currently supported?**
   NetPyNE is currently supported by grants from the NIH (U24EB028998), NSF (1904444-1042C), and the New York State Department of Health (DOH01-C32250GG-3450000).
3. **What is the plan for long term support?**
   The current funding for NetPyNE development will last until 2024. We currently have multiple other grants submitted and in preparation to support further development of NetPyNE. In the long-term, we plan to transition NetPyNE to community collaborative development and support.
4. **Are training and other supporting materials available?**
   Yes, there is a general reference (http://netpyne.org/reference.html), an API index (http://netpyne.org/package_index.html), tutorials (http://netpyne.org/tutorial.html), interactive Jupyter tutorials (https://github.com/Neurosim-lab/netpyne/tree/development/netpyne/tutorials), a YouTube channel (https://www.youtube.com/channel/UCSMxo4L1mfVgw6LgbFAxKTg), and a wide variety of examples (https://github.com/Neurosim-lab/netpyne/tree/development/examples).
5. **Any additional comments on sustainability and support**

Comparison

1. **Are there other similar SBP's available?**
   NetPyNE's standardized declarative language for high-level specifications is only currently comparable to NeuroMLLite (https://github.com/NeuroML/NeuroMLlite), which is currently under development.
   NetPyNE's data structure for network instances is comparable to that provided by NeuroML https://neuroml.org/, SONATA https://github.com/AllenInstitute/sonata, BMTK Bionet https://alleninstitute.github.io/bmtk/bionet.html and PyNN http://neuralensemble.org/PyNN/.
   NetPyNE's API or supporting tools/libraries are comparable to those provided by NeuroML, BMTK Bionet, and to a certain extent by other simulation tools such as NEST https://www.nest-simulator.org/, Brian https://briansimulator.org/, MOOSE https://moose.ncbs.res.in/ and GENESIS http://genesis-sim.org/.
2. **If yes, how do they compare on key INCF criteria?**
   NetPyNE's high-level declarative language for high-level specifications differs from NeuroMLLite in terms of the range of scales covered, from molecular up to large networks and extracellular space simulation – as opposed to NeuroMLLite, NetPyNE supports NEURON's Reaction-Diffusion (RxD) module (https://doi.org/10.3389/fninf.2013.00028). NetPyNE's declarative language also supports the definition of complex connectivity rules based pre- and post-synaptic cell spatial properties and using arbitrary mathematical expressions, as well as rules to distribute synapses across neuronal morphologies, which are not currency included in NeuroMLLite. Finally, there are robust and well tested tools to convert all components of NetPyNE's declarative language into NEURON network instances, and NetPyNE has been already been used to develop many models and interface with many tools (see sections above); whereas, NeuroMLLite is still under relatively early development and has not been officially released to the community for use.
   NetPyNE's data structure for network instances is similar to that of NeuroML, SONATA, Bionet and PyNN in that it stores components of the network in a hierarchical structure. It is particularly similar to BMTK Bionet, as both are designed specifically for NEURON models, and both include the NEURON objects within this network instance structure. NetPyNE network instance structure differs from the rest in that it also includes components at the molecular Reaction-Diffusion (RxD) scale.
   NetPyNE's API or supporting tools/libraries are unique in integrating many aspects of the modeling workflow under a single framework with a shared internal data format: (1) flexible, rule-based, high-level standardized specifications covering scales from molecule to cell to network; (2) efficient parallel simulation both on stand-alone computers and in high-performance computing (HPC) clusters; (3) automated data analysis and visualization (e.g. connectivity, neural activity, information theoretic analysis); (4) standardized input/output formats, importing of existing NEURON cell models, and conversion to/from NeuroML and SONATA; (5) automated parameter optimization across multiples scales (molecular to network) using grid search and evolutionary algorithms; (6) making all these features available programmatically or via a graphical user interface (GUI).
3. **Any additional comments on comparison with other SBP's?**