Integrating neuroinformatics tools in TheVirtualBrain

M. Marmaduke Woodman [1,*], Laurent Pezard [1,*], Lia Domide [3], Stuart Knock [1], Paula Sanz Leon [1], Jochen Mersmann [2], Anthony R. McIntosh [4] and Viktor Jirsa [1*]

[1] Institut de Neurosciences des Systèmes, 27, Bd. Jean Moulin, 13005, Marseille, France.
[3] Codemart, 13, Petofi Sandor, 400610, Cluj-Napoca, Romania.
[2] CodeBox GmbH, Hugo Eckener Str. 7, 70184 Stuttgart, Germany.
[4] Rotman Research Institute at Baycrest, Toronto, M6A 2E1, Ontario, Canada

ABSTRACT

TheVirtualBrain (TVB) is a neuroinformatics Python package representing the convergence of lines of work in clinical, systems, theoretical neuroscience in the integration, analysis, visualization and modeling of neural dynamics of the human brain as well as the imaging modalities through which these dynamics are measured. Specifically, TVB is composed of a flexible simulator for both neural dynamics and modalities such as MEG and fMRI, common analysis techniques such as wavelet decomposition and multiscale sample entropy, interactive visualizers for replaying cortical timeseries on the 3D surface or editing large-scale connectivity matrices, and an (optional) user interface accessible through modern web browsers. Tying together these pieces with persistent data storage, based on a combination of SQL & HDF5, is a rich, open-ended system of datatypes modeling (systems level) neuroscientific data and the relations among them. This data modeling system in parallel with the so-called adapter pattern architecture permit the integration of TVB with any other computational system, including MATLAB for which support is already available. Notably, TVB provides infrastructure for multiple projects and multiple users, possibly participating under multiple roles: a clinician may import diffusion spectrum imaging data, launch a tractography algorithm, and identify potential lesion points, and then share this data with a computational expert who would then enter to contribute simulation parameter sweeps and analyses, to test which lesion point is most probably given certain empirical imaging data, et cetera; this is one of many multi-user use cases supported by TVB. TVB also drives research forward on many levels: the simulator itself represents the systematization of several recent ad-hoc simulations in the modeling literature on human rest state. In these ways, TVB serves as an integrating platform for disparate expertises in the high level analysis and modeling of the human brain. This paper will begin with a brief outline of the history and motivation for TVB as a unified project *per se*. We proceed to describe the framework and simulator, giving usage examples in the web UI and in plain Python scripting. Finally, we compare TVB with the nearest neighbors in brain modeling, simulation performance, recent advances thereupon with native code compilation and GPUs, and the role of Python and its rich scientific ecosystem in TVB.

Keywords: connectivity, connectome, neural mass, neural field, time delays, full-brain network model, Python, virtual brain, large-scale simulation, web platform, GPUs

*to whom correspondence should be addressed: marmaduke.woodman@univ-amu.fr, viktor.jirsa@univ-amu.fr

1 MOTIVATION

While whole-brain level simulators have been developed and published for several years now, making the final step of connecting these simulations to empirical results has remained a challenge due to several factors:

1. Source code is typically not distributed, effectively closing the behavior, black box, etc.
2. The forward solutions required to obtain simulated M/EEG & fMRI data are non trivial, requiring interaction with several pieces of software
3. Published simulation methods for stochastic, delayed systems on surfaces are almost non existent
4. Managing all of the different computational pieces is typically challenging for those who work with empirical data

To address these concerns, a flexible architecture was developed to allow easy integration of any computational tools along with a system for describing typically types of data. A web based UI was developed for users not comfortable with programming, as well as MATLAB toolbox for interacting with the Python based framework, given that many neuroscientists are already comfortable with the MATLAB workflow. Lastly, a high performance, highly documented simulator along with various forward solutions have been implemented and released under a GPL licence to ensure universal access to high quality simulations, developed on the well-known Github, making it extremely easy for anyone to contribute.

1.1 Why another simulator?

This section actually motivates the integration story, otherwise nothing else is really necessary.

1.2 Why Python?

The core simulator actually began in MATLAB, however, as the needs expanded, the architecture, detailed in the next section, quickly outgrew the programming conventions typical of MATLAB packages.

Also, Python is free, and has an extremely rich ecosystem

1.3 Collaboration

In effect, we wish for a theoretician and clinician to be able to collaborate; to enable such a possibility, we require a platform to enable such opportunity.

1.4 Integration

Large scale simulation implies flexible integration. We shall see how this is enable by the architecture..

2 ARCHITECTURE

The architecture of *TheVirtualBrain* is divided into a framework and a scientific library.

2.1 Adapters

2.2 Datatypes

2.3 Et cetera . . .

$^{mw:}$ [I would like some help from Lia & Laurent to fill out this section]

3 USER INTERFACES

$^{mw:}$ [Focus more on scripting]

3.1 Web Interface

3.1.1 Projects, Users & Data

3.1.2 Simulator Interface

3.1.3 Visualizers & Analysis

3.1.4 Connectivity Tool

3.2 Python scripting

3.2.1 Hello Brain $^{mw:}$ [A quick, basic example]

3.2.2 Anatomy of a simulation $^{mw:}$ [Maybe cherrypick from the tutorial Stuart wrote]

3.2.3 Custom connectivity $^{mw:}$ [Perhaps you just want a simple network?]

3.2.4 Online feedback $^{mw:}$ [Or modify a parameter as a function of activity?]

4 SIMULATOR

The *TheVirtualBrain* simulator resembles popular neural network simulators in many fundamental ways, both mathematically and in terms of informatics structures, however we have found it necessary to introduce auxiliary concepts particularly useful in the modeling of large scale brain networks.

4.1 Node dynamics

In our models, nodes are no longer abstract neurons nor necessarily small groups thereof, but rather large populations of neurons, considered by, for example, the work of Wilson and Cowan.

$^{mw:}$ [quick list of models & their relevance]

4.2 Network structure

The nodes are embedded in two scale network structure. The first of which is derived from empirical measurements of the myelinated corticocortical connectivity, which we shall refer to as the inhomogeneous, large scale structure. An implication of this structure is the inherent delays in communication due to finite conduction velocity.

The second form of network structure is prescribed in the case of a cortical surface by the combination of said surface and a connectivity kernel. Together, these generate a homogeneous, local connectivity. For the current work, we consider this coupling to be instantaneous.

4.3 Integration of stochastic delay differential equations

Rather unlike other simulation paradigms, both noise and delays are common features of simulations in *TheVirtualBrain* . As such, the simulator is equipped with Heun and Euler methods for stochastic integration and the pairwise inter-regional delays are treated in an efficient fashion.

$^{mw:}$ [the general equation here]

$^{mw:}$ [Describe handling of delays?]

4.4 Forward solutions

One of the primary goals of the *TheVirtualBrain* simulator is to allow for the simulation of empirical whole-brain data, such as EEG or fMRI. *TheVirtualBrain* implements several forward solutions as so-called monitors, including MEG, sEEG, EEG and fMRI.

Crucially, given the amount of data *TheVirtualBrain* may produce, especially for simulations on a cortical surface, each of the monitors is "on-line" in the sense that is runs in constant space.

4.5 Native code generation for C & GPU

Several of the core components (integrators, mass models, coupling functions) have targeted towards a C source code backend, which has allowed for the compilation of simulations to native code loaded either as a shared library accessed via the `ctypes` modules or as CUDA kernels accessed via the `pycuda` module **?**. While such an approach may provide speed ups, they depend on the presence of a C compiler and, in the case of GPU, the CUDA toolkit and a compatible graphics card.

4.6 Other simulators compared to *TheVirtualBrain*

5 FUTURE WORK

On going work on the simulator includes

1. Code generation in the simulator to compile specifications to native code running independently of TVB or on the GPU.
2. MATLAB interface
3. Spatial derivatiaves (?)

As previously stated, the primary goal of the *TheVirtualBrain* is to provide the means to pass all the way from structural imaging data to dynamic imaging. We aren't there yet, as several pieces still require scientific consensus and standards. Notably,

1. DTI pipeline
2. Connectome Project

3. Structural imaging processing via FreeSurfer (pySurfer, NiPy, etc.)
4. NeuroML project

In the near future, our informatics efforts outside of these areas will focus on making the platform accessible and demonstrating the validity of the simulations with respect to empirical data.