

Reassembly and Interfacing Neural Models Registered on Biological Model Databases

Mihoko Otake^{1,2}

otake@cb.k.u-tokyo.ac.jp

Toshihisa Takagi^{1,3}

tt@k.u-tokyo.ac.jp

¹ Science Integration Program - Humans, Department of Frontier Sciences and Science Integration, Division of Project Coordination, The University of Tokyo, Kashiwa-no-ha 5-1-5, Kashiwa-shi, 277-8568 Japan

² PRESTO program, Japan Science and Technology Agency

³ Department of Computational Biology, Graduate School of Frontier Science, The University of Tokyo, Kashiwa-no-ha 5-1-5, Kashiwa-shi, 277-8568 Japan

Abstract

The importance of modeling and simulation of biological process is growing for further understanding of living systems at all scales from molecular to cellular, organic, and individuals. In the field of neuroscience, there are so called platform simulators, the de-facto standard neural simulators. More than a hundred neural models are registered on the model database. These models are executable in corresponding simulation environments. But usability of the registered models is not sufficient. In order to make use of the model, the users have to identify the input, output and internal state variables and parameters of the models. The roles and units of each variable and parameter are not explicitly defined in the model files. These are suggested implicitly in the papers where the simulation results are demonstrated. In this study, we propose a novel method of reassembly and interfacing models registered on biological model database. The method was applied to the neural models registered on one of the typical biological model database, ModelDB. The results are described in detail with the hippocampal pyramidal neuron model. The model is executable in NEURON simulator environment, which demonstrates that somatic EPSP amplitude is independent of synapse location. Input and output parameters and variables were identified successfully, and the results of the simulation were recorded in the organized form with annotations.

Keywords: modeling, simulation, biological database, biological process, neuroinformatics

1 Introduction

It is widely recognized that sharing primary data is very important for the progress of life science. In the field of genomics and proteomics, individual scientists routinely contribute their sequence data to centralized databases at the time of publication. As a result, the databases are growing daily, and efforts for integrating biological data at molecular level have been done in ever more sophisticated ways [9, 10]. In the field of neuroscience, the Human Brain Project (HBP), a multi-agency US government initiative established in 1993 has been supported research into databases and related tools for neuroscientists, in parallel with the Human Genome Project (HGP) [14]. In 2005, seven member countries of the OECD's Global Science Forum have launched a project, the International Neuroinformatics Coordinating Facility (INCF), to promote international collaboration among scientists and create new ways of sharing and analyzing data to improve our understanding of how the human brain works. The datasets generated by neuroscientists are far more diverse and complex compared to the genomic sequence data. The relevant variables include morphology, functional connectivity, neurophysiology, chemistry, molecular biology, genomics, brain imaging and behavior [11]. Various databases have been developed which deal with data at all scales [3]. Therefore, the method for integrating these data

and knowledge is indispensable, which will provide deeper understandings of how the nervous systems work in both health and disease.

The purpose of this study is to develop the neural simulator that can calculate both microscopic and macroscopic states of the nervous system. Characteristics of the nervous system should be modeled at all scales in order to develop such simulators. Fortunately, a variety of neural models are registered on neural model databases, which are created and reviewed by neuroscientists. We should be able to develop a simulator which is composed of reliable partial models, if we take advantage of them. However, reusing these models requires much time and labor, because the variables and parameters are not annotated. Also, each model doesn't consider connectivity to other models since the models are originally developed for particular experiments. In this paper, we propose a novel method for reassembly and interfacing models registered on biological model databases in order to integrate these models.

2 Neural Models and Model Databases

2.1 Model Database

Modeling is one of the powerful methods for integrating the new information into existing hypothesis or formulating new ones, and they carrying out additional experiments to test these hypothesis [15, 11]. The models sort relevant data from the database, and help the researcher spot inconsistencies. They are formal representations of hypothesis, which provide a way for life scientists to communicate and compare their ideas quantitatively. Model databases deal with such models and diverse data for building the models.

In the field of neuroscience, Bower's group has developed model database named ChannelDB [1] for GENESIS [2] simulation environment. Usui *et al.* [16] have developed Visiome Platform that provides access to reusable resources in the research field of vision science including mathematical models, experimental data, analysis libraries and related information. Shepherd and his colleagues have developed and maintain the model database named ModelDB [13]. ModelDB is well organized, which provides an accessible location for storing and efficiently retrieving computational neuron models. Users of ModelDB can search for models by author name; a particular neuron type; containing a particular property such as currents, receptors, or transmitters; related topics, e.g. synaptic plasticity, pattern recognition; simulation environment. Model code can be viewed before downloading and browsers can be set to auto-launch the models. There are pointers to the primary literatures for each model. Models can be coded in any language for any environment, though ModelDB has been initially constructed for use with NEURON [7] and GENESIS. Over a hundred NEURON models are available on the database.

In this study, we focus on NEURON models registered on ModelDB and propose method for reassembly and interfacing the models in order to mine, reanalyze and integrate the models and underlying knowledge behind the models.

2.2 Neural Models and Their Simulation Environments

For the purpose of pointing out the difficulty of the model integration, we briefly explain the neural simulation platforms and the neural models. In the field of neuroscience, there are so called platform simulators, the de-facto standard neural simulators. NEURON [7] and GENESIS [2] are typical platform neural simulators for models of individual neurons and networks of neurons that are closely linked to experimental data. They can simulate individual neurons and networks of neurons with properties that may include, but are not limited to, complex branching morphology, multiple channel types, inhomogeneous channel distribution, ionic diffusion, and the effects of second messengers. They provide tools for constructing, exercising, and managing models, which are widely accepted among

experimentalists. They have been used in research reported in more than two hundreds scientific articles respectively.

The principal components of the simulation system and the various modes of interacting with these simulators are: script language interpreter, graphical user interface (GUI), simulation engine including those for input/output and for the numerical solution of the differential equations, data files and pre-compiled object libraries. They use a high-level simulation language to construct neurons and their networks. Commands may be issued either interactively to a command prompt, by use of simulation scripts, or through the GUI. A particular simulation is set up by writing a sequence of commands in the scripting language that creates the model itself and the GUI for a particular simulation. The scripting language and the modules are powerful enough that only a few lines of script are needed to specify a sophisticated simulation.

2.3 Usability Problems of Registered Models for Integration

There are two major problems for utilizing and integrating the registered models.

1. The input, output and internal state variables and parameters of the models are not explicitly defined. Users have to identify them by executing the models and reading through the papers in which the simulation results are described in detail.
2. The models are designed on the assumption that they are executed stand alone. Each model has its own GUI with tunable parameters.

Models registered on the model database are contributed by neuroscience researchers. The simulation results of the models are demonstrated in the peer reviewed papers. The models themselves are also reviewed by the database curators with minimal modifications. The validity of the models is guaranteed. However, the models do not contain detailed description of the variables and parameters. This is the first problem. It is not clear which represents input, output, or internal state of the neurons. It is no wonder if we think of the origin of the models. The models were originally developed in order to solve the particular scientific problems. They are not intended to be used by other researchers as building blocks for the integrated models. Therefore, the researchers who would like to use the models have to identify the meaning of the variables and parameters by executing the models, analyzing the simulation results, and reading through the papers.

The second problem is the side effect of following two features which derive flexibility and convenience of the simulators. The first feature is a GUI that can be used to create models, run initial exploratory simulations, set parameters, control common voltage and current stimuli, and graph variables as functions of time and position. The second feature is an interpreter that provides a complete programming language which is useful for customization of the GUI, advanced data analysis, and optimization. Because of these two features, data, logics and interfaces are combined. Thus, models should be organized to improve usability for integration.

3 Reassembly and Interfacing Method

In this study, we propose method for reassembly and interfacing neural models registered on the model database. It is an extremely important step toward understanding the integrated function of the nervous system. The basic concept of the method is listed below.

1. Input and output variables and parameters should be identified systematically with minimal efforts.
2. Modification of the model should be kept minimal.

3. Simulation environments are not reconfigured.

The first strategy comes from the fact that there are over a hundred models on the database, which makes a long time to read through all files constituting the models. Therefore, we propose efficient extraction rule for identifying input/output (I/O) variables and parameters. The second one considers that the models are already reviewed and guaranteed the standalone operation. This advantage should be reserved. We add some small script to the model files for I/O. The third is based on the fact that the simulation environments are always updated. If we modify the simulators themselves, we have to rewrite them when the newer versions are released. We develop interface programs without changing the simulator. The proposed method contains following three phases which correspond to these strategies.

3.1 Identification of Input and Output of the Models

Extracting the Plot Commands from the Model Files

The models are constituted by a sequence of commands in the scripting language that creates the model itself and the GUI for a particular simulation. They contain vast numbers of variables and parameters. Some are only for internal use; others are used for plotting the graph. Each has its own range of value. If the model is used in not assumed conditions, the obtained results may be nonexistent. Basically, the GUI of the models is disturbance for integration. Conversely, if we draw attention to the variables and parameters which are plotted on the graphs, we can see that they are candidates for input and output of the models. We can also find some clues for their ranges. Therefore, the first step for identifying the input and output is to extract graph plotting commands from the model files. Retrieval terms for NEURON simulator are *.addvar* and *.plot*. The variables which are plotted on the graph are candidate for output. The parameters which increment in the repetitive plotting process are candidates for input.

Extracting the Units and Meaning from the Graphs in the Papers

The simulation results of the registered models are demonstrated in the peer reviewed papers. The corresponding graphs can be found from these papers. The units and meaning of the variables and parameters are illustrated there. They are used for annotation. This step is not automatic but manually carried out.

3.2 Reassembly of the Models

Definition of Input and Output in UDF format

Now that the input and output variables and parameters are identified, they should be stored and recorded in a easily readable and understandable manner. In this study, the user definable format (UDF) is employed for data representation of the input and output, because it provides quite simple representation of experimental data [5] where overhead is kept minimum compared to XML. The format was originally developed for integrated simulator platform named GOURMET of OCTA project [17] for simulation of soft materials [5]. GOURMET has not been used other than the simulation of soft materials, although it is potentially applicable to any kinds of simulation. UDF is a plain text file, and consists of definition part and data part. In the definition part, you can define the data structure of your engine in a simple and natural way. Data are arranged in the data part according to the structure. The specific data will be explained in detail in the experiment section.

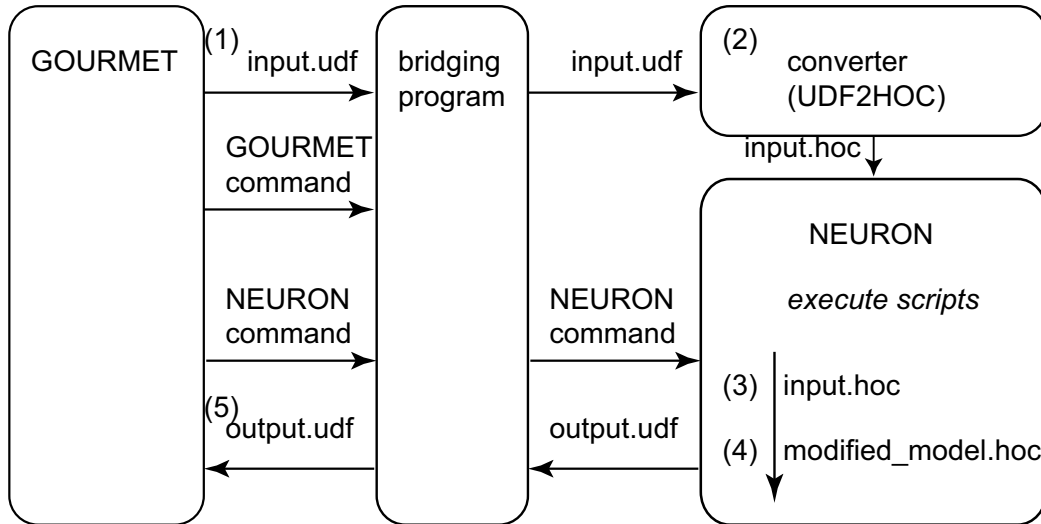


Figure 1: System structure of the integrated simulation environment. It comprises integrative simulation platform (GOURMET), neural simulator (NEURON), conversion script (UDF2HOC), and the bridging program.

Conversion of Input and Output

The simulators cannot deal with UDF format. In order to get over the input, the program named UDF2HOC was implemented which converts UDF format into simulation script format. In the case of NEURON, the simulation scripting language is HOC. The program was executed before loading the models in the neural simulator. Then, a few lines were added in the model script files in order to output the simulation results into UDF format.

3.3 Interfacing the Models

The integrated simulation platform was considered for connecting different kinds of models which run on different simulation environments. In this study, GOURMET was selected as simulation platform on which neural simulation programs run, because it provides a common GUI. GOURMET is not only a place for various simulation programs to meet and exchange the information they have, but also an editor of the input data, a viewer of the output data, a tool to make graphs and animations. The users can easily verify the simulation results because of these characteristics.

Programs which are connected to GOURMET should be able to send and receive UDF files and connected simulator commands, to receive GOURMET commands. Thus, we developed bridging program with these functions, which connects GOURMET and typical neural simulator, NEURON. In this way, we can interface the neural simulators without modifying their source codes. The operation of the bridging program is described below, which illustrated in Figure 1.

1. Input data in UDF format (input.udf) is sent from GOURMET to the bridging program.
2. The bridging program submits the input data to the converter (UDF2HOC) and executes it. The converter generates the HOC format file (input.hoc) from input data, which can be interpreted in the NEURON simulation environment.
3. NEURON simulator executes the generated hoc file.
4. Then, NEURON runs the modified neural model so that the model generates output in UDF format.

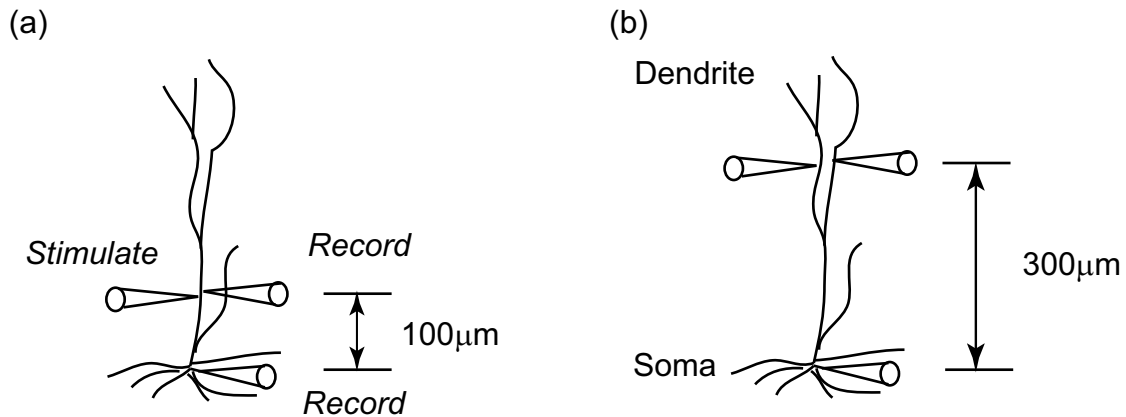


Figure 2: Experimental apparatus of the hippocampal pyramidal neural cell model. EPSPs were recorded for (a) a proximal ($100\ \mu\text{m}$ from the soma) and (b) more distal ($300\ \mu\text{m}$ from the soma) synaptic inputs.

5. The result (output.udf) is sent from NEURON to GOURMET through bridging program.

4 Experiment

4.1 Hippocampal Pyramidal Cell

We applied the method to the models registered on ModelDB. The hippocampal neural cell model is one of them on ModelDB, whose result of application of the method is described in detail hereafter. The model demonstrates that somatic EPSP amplitude is independent of synapse location in hippocampal pyramidal neurons [12]. The author of the model characterized the amplitude and kinetics of excitatory synaptic input across the apical dendrites of CA1 pyramidal neurons using dual whole-cell recordings. They found that dendritic EPSP amplitude increases with distance from the soma, counterbalancing the filtering effects of the dendrites and reducing the location dependence of somatic EPSP amplitude. The distance between the site of input (dendrite) and soma were altered for both experiments and simulations. EPSPs were recorded for a proximal ($100\ \mu\text{m}$ from the soma) and more distal ($300\ \mu\text{m}$ from the soma) synaptic inputs (Figure 2). The results suggest that a progressive increase in synaptic conductance seems to be primarily responsible for normalizing the amplitudes of individual inputs.

4.2 Results

Identification of Input and Output of the Model

The following three lines were extracted from one of the script files of the hippocampal pyramidal cell model, “main_example_epsps.hoc”.

```
g2[synapse_pos_cntr].plot(t)
g2[synapse_pos_cntr].addvar("s.v(.5)",1,1)
g2[synapse_pos_cntr].addvar("d[i].v(.5)",2,1)
```

We can estimate that the horizontal axis of the graph is “t”, while the vertical axis of the graph are “s.v(.5)” and “d[i].v(.5)”. The graph figure of the paper [12] defines that the former one is the EPSP recorded at the soma while the latter one is the EPSP recorded at the dendrite. Therefore, the output variables of the model are time “t”, EPSP at the soma “s.v(.5)” and EPSP at the dendrite “d[i].v(.5)”. The unit of the horizontal axis is [ms] while the vertical axis is [mV].

Then, a comment was found in the file on “i”.

```
// i is dendritic compartment for synapse
```

The values “38, 153” were substituted for the parameter “i”. They are reference numbers of the compartment of synapse. The number suggests the distance from the soma to the injection site. The former compartment should be the proximal site (100 μm from the soma), while the latter one should be the distal site (300 μm from the soma), through comparison of the simulation results and the graph in the paper. The input parameter of the model is “i”, the reference number of the compartments. It took thirty minutes for manual identification, once the commands were automatically extracted.

Reassembly of the Models

In summary, the output variables of the model are time “t”, EPSP at the soma “s.v(.5)” and EPSP at the dendrite “d[i].v(.5)”. The unit of the horizontal axis is [ms] while the vertical axis is [mV]. The input parameter of the model is “i”, the reference number of the compartments. There is no unit for the input. Since the input contains only one value, we describe the output data of the modified model. The definition part of the output was defined as:

```
\begin{global_def}
  expresult[]:{
  t: double [ms] "time"
  sv: double [mV] "somatic voltage"
  dv: double [mV] "dendritic voltage"
  }
\end{global_def} .
```

The first items before the “:” are variables, the second items after the “:” are data types, words in parentheses are units and the final items are the annotations of the variables. The commands for the output were added in the script files. The data part of the output was:

```
\begin{data}
  expresult[]:[
  {0.05 -64.9993 -64.99}
  {0.1 -64.9974 -64.9689}
  {0.15 -64.9944 -64.9458}
  (snip)
  {39.9 -64.9951 -64.9951}
  {39.95 -64.9951 -64.9951}
  {40 -64.9952 -64.9952}
  ]
\end{data} .
```

The set of time, somatic voltage, dendritic voltage data constitutes the data part.

Interfacing the Models

The model was loaded and executed in NEURON simulator, which received the commands from the simulation platform GOURMET, through the bridging program. Figure 3 shows the viewer of GOURMET which displays the simulation result. In Figure 3 (a), we can see that the output data in UDF format is represented in the tree structure and table. The data is also plotted on the graph (Figure 3 (b)). The external program successfully controlled NEURON simulator and the results of the simulation were recorded in the organized form.

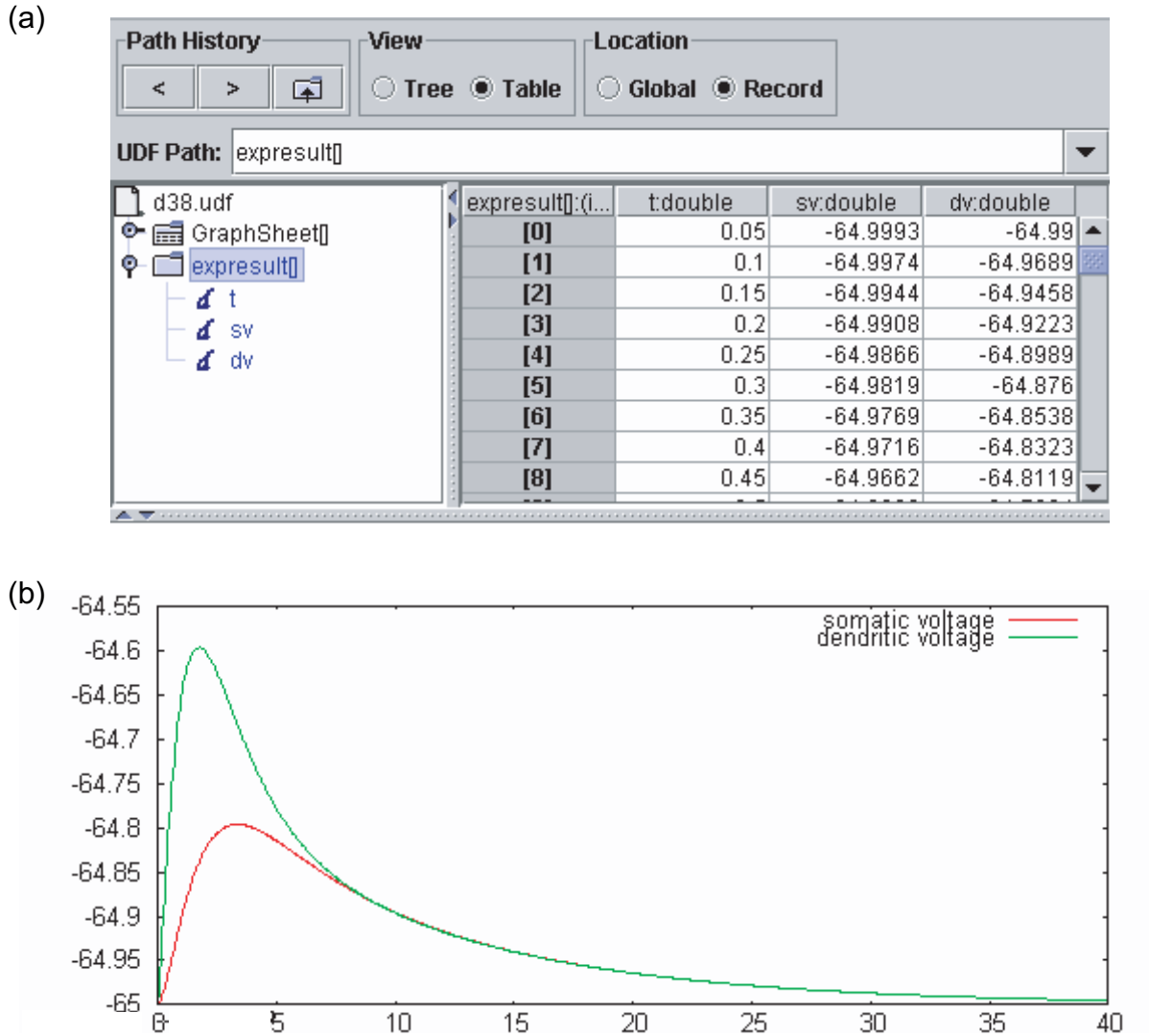


Figure 3: Simulation results which are displayed in integrated simulation platform, GOURMET. (a) The output data in UDF format is represented in the tree structure and table. (b) The output data is plotted on the graph.

5 Discussion

In this section, we discuss limitations and applications of the proposed method.

5.1 Selection of the Data Format

We selected user definable format (UDF) because of its simplicity and flexibility. We would like to point out that annotation of variables and parameters of the data and models are very important for sharing the knowledge embedded in the model in order to integrate the models. UDF provides simple interface for annotation. Data format based on XML for life sciences such as SBML [8] are commonly available. NeuroML [6] is one of the candidates for the representation of neurophysiological data. Since the UDF data are simple, they can easily converted to other data format such as NeuroML. More importantly, the models in UDF format are incorporable to the models of other scales, where standard formats are different or unavailable.

5.2 How to Extend the Flexibility of the Interface

Amplitude of the current injection was fixed while the position of current injection was altered in the hippocampal pyramidal neuron model, which we selected for description. The output of the model was EPSP, while other variables such as time series of spike signals were ignored. We will hit upon the problem if the model is connected to different neural models of SpikeNET [4] which require spike input. Input and output variables and parameters should be added for the purpose of enhancing the flexibility of the model connection. Standardization of the parameters and variables is needed. Since the interface of the neural cell is relatively clear, we can reduce the numbers of input and output. Identification of input and output may not be clear for other kinds of cellular models compared to neural models. Further investigation is needed for other biological models. They are to be solved as a future work.

5.3 How to Apply the Method to Different Simulation Environment

In this paper, we have described the method with one of the typical neural simulation environment, NEURON. Our technique applies to the models of different simulation environment if we alter the extraction keyword. If the extraction keywords of NEURON are replaced to those of GENESIS, the method is applicable to the models of GENESIS. The bridging program should be implemented for each simulation environment.

6 Conclusion

We have proposed a novel method with three phases that reassembles and interfaces the models which are registered on biological model databases.

- The first phase identifies the input and output parameters and variables through extracting the plot commands. Units and annotations of these input and output are found in the corresponding papers.
- The second phase organizes the input and output data in user definable format (UDF).
- The third phase interfaces the model and simulation environment to send and receive external commands and data.

The method was applied to the neural models for NEURON simulator on one of the typical neural model database named ModelDB. We succeeded in obtaining incorporable neural models and annotated data of simulation results. This work proposes original approach among various efforts toward integrating the life science as well as neuroscience data and knowledge. Future work includes providing these improved models available on the web and building larger scale models with them. Models that work in the simulation environments other than NEURON will also be interfaced to the simulation platform for integrating the models independent of simulation environments. Enhanced integrative simulation platform will be implemented for flexible conjunction of different simulation environments.

Acknowledgments

This work is supported by Japan Science and Technology Agency Grant for PRESTO program “Development of the Bilateral Multiscale Neural Simulator” (PI: Mihoko Otake) and a Grant-in-Aid for Scientific Research on priority area Systems Genomics (#014) from the Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT). The authors express deep appreciation to Prof. M. Doi for comprehensive introduction to UDF technology, and Prof. H. Komiyama for his suggestions and encouragements.

References

- [1] Beeman, D. and Bower, J.M., Simulator-independent representation of ionic conductance models with ChannelDB, *Neurocomputing*, 58–60:1085–1090, 2004.
- [2] Bower, J.M., Beeman, D., and Hucka, M., The GENESIS simulation System, in Arbib, M.A. (ed.), *The Handbook of Brain Theory and Neural Networks, 2nd edition*, Cambridge, MA, MIT Press, 475–478, 2003.
- [3] Chicurel, M., Databasing the brain, *Nature*, 406:822–825, 2000.
- [4] Delorme, A. and Thorpe, S., SpikeNET: An event-driven simulation package for modeling large networks of spiking neurons, *Network: Computation in Neural Systems*, 14:613–627, 2003.
- [5] Doi, M., Challenge in polymer physics, *Pure Appl. Chem.*, 75:1359–1615, 2003.
- [6] Goddard, N.H., Hucka, M., Howell, F., Cornelis, H., Shankar, K., and Beeman, D., Towards NeuroML: Model description methods for collaborative modelling in neuroscience, *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, 356:1209–1228, 2001.
- [7] Hines, M.L. and Carnevale, N.T., The NEURON simulation environment, *Neural Computation*, 9:1179–1209, 1997.
- [8] Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., *et al.*, The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models, *Bioinformatics*, 19:524–531, 2003.
- [9] Joshi-Tope, G., Gillespie, M., Vastrik, I., D’Eustachio, P., Schmidt, E., de Bono, B., *et al.*, Reactome: A knowledgebase of biological pathways, *Nucleic Acids Res.*, 33:D428–D432, 2005.
- [10] Kanehisa, M. and Goto, S., KEGG: Kyoto encyclopedia of genes and genomes, *Nucleic Acids Res.*, 28:27–30, 2000.
- [11] Koslow, S.H., Should the neuroscience community make a paradigm shift to sharing primary data?, *Nat. Neurosci.*, 3:863–865, 2000.
- [12] Magee, J.C. and Cook, E.P., Somatic EPSP amplitude is independent of synapse location in hippocampal pyramidal neurons, *Nat. Neurosci.*, 3:895–903, 2000.
- [13] Migliore, M., Morse, T.M., Davison, A.P., Marenco, L., Shepherd, G.M., and Hines, M.L., ModelDB: Making models publicly accessible to support computational neuroscience, *Neuroinformatics*, 1:135–139, 2003.
- [14] Shepherd, G.M., Mirsky, J.S., Healy, M.D., and Singer, M.S., The human brain project: Neuroinformatics tools for integrating, searching and modeling multidisciplinary neuroscience data, *Trends Neurosci.*, 21:460–468, 1998.
- [15] Tomita, M., Hashimoto, K., Takahashi, K., and Shimizu, T.S., E-CELL: Software environment for whole-cell simulation, *Bioinformatics*, 15:72–84, 1999.
- [16] Usui, S., Visiome: Neuroinformatics research in vision project, *Neural Netw.*, 16:1293–1300, 2003.
- [17] <http://octa.jp/>