1st INCF Workshop

on

Large-scale Modeling of the Nervous System

December 12–13, 2006 - Stockholm, Sweden

incf International Neuroinformatics Coordinating Facility

www.incf.org

1st INCF Workshop on Large-scale Modeling of the Nervous System

December 12–13, 2006

International Neuroinformatics Coordinating Facility Secretariat

Stockholm, Sweden

Authors

Mikael Djurfeldt and Anders Lansner

Scientific Organizer

Anders Lansner, KTH, Stockholm, Sweden

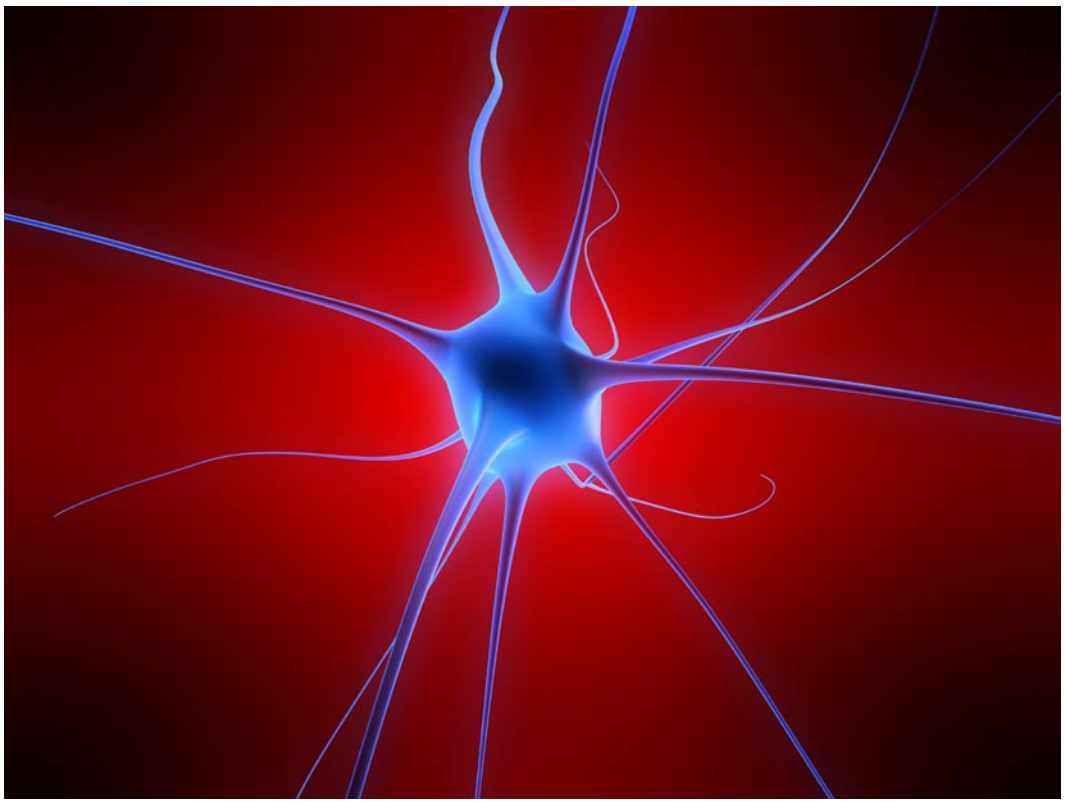
Workshop Participants

David Beeman, University of Colorado at Boulder, Boulder, CO, USA
Andrew Davison, UNIC, Gif-sur-Yvette, France
Markus Diesmann, RIKEN Brain Science Institute, Wako, Japan
Rodney Douglas, UNI–ETH, Zürich, Switzerland
Jens Eberhard, Ruprecht-Karls-University of Heidelberg, Heidelberg, Germany
Frederick Harris, University of Nevada, Reno, NV, USA
Michael Hines, Yale University, New Haven, CT, USA
Thomas Natschläger, Software Competence Center Hagenberg, Hagenberg, Austria
Charles Peck, IBM Research Division, Yorktown Heights, NY, USA
Shiro Usui, RIKEN Brain Science Institute, Wako, Japan
Gabriel Wittum, Ruprecht-Karls-University of Heidelberg, Heidelberg, Germany
Rapporteur: Mikael Djurfeldt, KTH, Stockholm, Sweden

Supported by the INCF Central Fund and the Swedish Foundation for Strategic Research

Contents

Contents



1 Executive Summary

The goal of this workshop was to survey current demands, ongoing activities, and plans relating to development of tools for scalable neural network simulation. Areas covered included software components for preprocessing/model setup, as well as for storage, analysis, and visualization of results. Participants discussed the need for coordinated action in the field with regard to model, method and tool development.

In the following summary, we present three major findings and seven recommendations developed by workshop participants.

Findings

The workshop participants reported:

- **on difficulties in reproducing simulations from published articles**
- **that the current diversity of simulators creates vigor in the field and has benefits for the validation of models**
- **on the importance of facilitating software interoperability and re-use of simulation software components**

Recommendations

The following recommendations were made by the workshop participants:

- **Arrange an annual workshop on large-scale modeling with the aim of developing standards for improving reusability of software.** A workshop in the same format as the Telluride Workshops on neuromorphic engineering would give an opportunity to teach large-scale simulation technology while simultaneously presenting an optimal environment for practitioners and students to discuss, test, and evaluate approaches to software interoperability.
- **Implement a standard simulator test suite.** 1. INCF should promote development of a set of simulator "benchmarks" for the purposes of verifying correctness of computation and serving as a standard for simulator performance. These should preferably be selected with respect to published models not contrived by the simulator developers. 2. INCF should coordinate and contribute to raising funding for the implementation of the test suite by members of the labs that have developed the simulators. 3. INCF should maintain web pages with the results of the test suite for each simulator. These pages should be updateable so that old information can be superseded by current best practices in each simulator.
- **Implement an experimental framework for connecting software components.** A feasibility study should be performed regarding the possibility of on-line communication between different software modules, for example two parallel simulators. INCF should allocate resources for implementing a software library with the communication interface.
- **Develop publication guidelines for ensuring the reproducibility of simulations.** INCF should develop guidelines for publications with computer simulations and publish these on the INCF homepage as a reference for authors. INCF should also encourage publishers to implement policies for making models available in conjunction with publication.
- **Encourage, support and fund work on method development within computational neuroscience.** INCF should inform funding agencies about the need for research on methods in computational neuroscience, and on methods for large-scale simulation in particular.
- **Encourage, support and fund the production of publications on concepts and techniques for large-scale simulations.** INCF should inform publishers about the need for published work on methods within the field of large-scale simulations and computational neuroscience in general, and about the current growth in this area.
- **Continue work on defining areas in need of support and coordination.** Workshop participants agreed that the 1st INCF Workshop on Large-scale Modeling of the Nervous System was successful. However, there is still a great need for further work on coordination within the field. INCF should therefore arrange further workshops with the aim of defining areas in need of support and coordination.

2 Introduction

Computer simulation is an increasingly important tool in neuroscience research. As computational modeling techniques become integrated with experimental neuroscience, more knowledge can be extracted from existing experimental data. Quantitative models help to explain experimental observations and seemingly unrelated phenomena. They assist in generating experimentally testable hypotheses and in selecting informative experiments.

Supercomputers are becoming ever-more powerful. It has even become possible to simulate models of a scale corresponding to substantial parts of a small mammal's neocortex with biologically detailed compartmental neuron models. At the same time, recent progress in experimental techniques holds the promise of supplying modelers with data of an unprecedented level of detail. Large-scale models are becoming an essential tool in bridging multiple levels of organization in the description and understanding of the nervous system. However, large-scale modeling brings many new challenges, and there is a sense in the community that we could benefit from coordination efforts in areas such as standardization, interoperability and verification.

The purpose of this report is to summarize discussions and recommendations of the workshop and to answer the question of how INCF can support the large-scale modeling community. To this end, section 4 surveys the current status of large-scale modeling in neuroscience and the scientific needs. Section 5 reviews the status of current tools for large-scale modeling and discusses topics in need of development with regard to these tools. Finally, section 6 discusses how INCF can support the community directly or contribute to changes with regard to other organizations and scientific infrastructure.

3 Concepts

In the following, some basic concepts with regard to modeling and simulation of the nervous system will be restated and clarified. This will provide a basis for a discussion of the various current developments in the field.

3.1 What do we mean by "model"?

Mathematical models are the language of science. According to *Wikipedia*, a *mathematical model* is an *abstract model* expressed in mathematical language. Further:

An abstract model is a theoretical construct that represents something, with a set of variables and a set of logical and quantitative relationships between them. Models in this sense are constructed to enable reasoning within an idealized logical framework and are an important component of scientific theories. Idealized here means that the model may make explicit assumptions that are known to be false (or incomplete) in some detail. Such assumptions may be justified on the grounds that they simplify the model while, at the same time, allowing the production of acceptably accurate solutions.

It should be remembered that a mathematical model of reality should *always* be regarded as idealized in the sense above. At least this holds true for all types of model considered in this report.

3.2 Model complexity

The golden standard with regard to model building is well captured by words often attributed to a certain famous physicist: "Everything should be made as simple as possible, but not simpler." Model complexity can be measured in many ways. For this discussion, two measures are particularly important: 1. the number of model parameters, and 2. the number of state variables in the model, which will here be denoted as *model dimension*.

A model is always tailored to answer a specific set of questions. For a physicist, the ideal is to pick the model with the smallest number of parameters which will still be sufficient to answer the scientific questions posed. This has several good consequences:

1. A simple model can be analyzed and understood. More complex aspects of nature can be understood through the strategy of divide-and-conquer.

2. A simpler model expresses a simpler scientific hypothesis in the sense of Occam's razor. It cuts away elements that are irrelevant to the problem studied.

3. A model with fewer parameters is better constrained, making it easier to falsify. A model with many parameters can easily be adapted to the results of any experiment.

Of the two dimensions of simplicity mentioned, the first one is most important. A model which has few parameters is more likely to express a well defined piece of scientific knowledge in the form of a strong, i.e. falsifiable, hypothesis, while the lack of this kind of simplicity threatens all three of the benefits above. A large model dimension can make the model harder to analyze and understand but there are many techniques available for handling this kind of complexity.

In addition, it is important to make a distinction between *free* parameters—parameters which are tuned by the modeler or software to achieve a certain model behavior—and parameters constrained by experimental data. A model should have few free parameters in order not to lose benefit 3 above. Also, all model behavior which is used to tune parameters is transformed from an "output" of the model to an "input", i.e., it can no longer be claimed as a result, or prediction, of the model.

3.3 Abstraction and level of description

The tool used to achieve a simple model is abstraction. The system is described at a certain level and elements which are not believed to be important for answering the scientific questions asked are taken away.

Neuroscience spans many levels of description from molecules to behavior. But what does *level* mean? Churchland and Sejnowski (1992) discuss three categories of levels. The structure of the nervous system has many different spatial scales with substructures such as molecules, synapses, neurons, microcircuits, networks, regions and systems (list slightly modified from Churchland). We call these *levels of organization*. With some good will, *behavior* could be added at the top of this hierarchy.

Marr (1982) described levels along a different dimension in his *levels of analysis*: 1. the computational level of abstract problem analysis, 2. the algorithmic level, specifying a formal procedure to perform the task so that a given input will yield the correct output, 3. the level of physical implementation. Marr argued that a higher-level question was largely independent of the levels below and could be analyzed independently of the lower level. However, it should be noted that Marr used, to a large extent, neurobiological considerations to constrain and inspire his computational theories and algorithms.

Churchlands third category, *levels of processing*, will not be discussed here.

For a model of the primate primary visual cortex, V1, Marr's computational level would correspond to *what* computations are being performed in V1 and *why*. The algorithmic level would correspond to *how* the information being processed on the computational level is represented and how the computations are carried out, while the level of physical implementation describes the actual computational elements performing these computations.

A typical large-scale network model, with single- or multicompartment units, thus belongs to the level of physical implementation, since it deals with neurons and synapses. But it can still be inspired, and constrained, from the other two levels, and can embody principles from the "higher" levels.

So far, we have discussed levels of organization and levels of analysis. An *abstract model* leaves out aspects of the description of reality in order to achieve simplicity. Sometimes this means leaving out elements from a lower level of organization, but it can also mean describing something at a higher level of analysis. A model of visual processing in terms of filter banks and kernels is considered more abstract than a model of neuronal populations in V1. Thus, we also talk about level of abstraction.

3.4 Detailed versus abstract models

A detailed model is often considered the opposite of an abstract model. In this report, a *detailed model* is defined to mean a model which spans several levels of organization. A model which spans the levels from networks to behavior can thus simultaneously be more abstract and more detailed compared to a model restricted to a single but lower level of organization. A model of brain imaging data which incorporates networks of simple units can be more detailed than a statistical model of an ion channel. Yet, because it leaves out so much of the detail beneath the level of networks, it can also be more abstract than the latter model.

3.5 Realism

In order to say something about reality, and in order for a corresponding hypothesis to be falsifiable, a model needs to be well rooted in empirical data, i.e. formulated in a way that is consistent with a large set of experimental data.

It should now be made clear that, just as a model can be formulated on all of the levels of organization discussed in sec-

tion 3.3, empirical data can be obtained on different levels of organization. It should also be pointed out that models can be formulated on higher levels of analysis (in the sense of section 3.3): A cognitive psychologist can retrieve behavioral data and construct models at the algorithmic level, without a direct reference to how these processes are physically implemented in the brain. Brain imaging retrieves data at the levels of networks, regions and systems. Electrophysiology collects data at many levels of organization and this data can be used to construct models at the level of physical implementation.

A model based on data from a lower level of organization, or formulated in terms of elements from multiple lower levels of organization, is not necessarily more *realistic* than data formulated and rooted at a higher level. Rather, a model is realistic if it is well rooted in empirical data at the given level, if its parameters are well constrained by these data, and if it *correctly predicts* data which has not been used to tune the model.

Not only the parameters of a model, but also the equations, represent assumptions about reality. Some models are based on *first principles*, that is, their equations are established laws of physics. Models based on first principles, or models with equations which can be derived from the laws of physics, can be trusted to a larger extent than other models, because they rely less than other models on assumptions in the form of peculiarities of the model equations or fitting of parameters.

4 Directions in modeling of the nervous system - scientific needs

4.1 The role of the model in current neuroscience

This section contains a brief discussion of the various roles that models currently have in neuroscience. As stated above, models are the language of science. We use models to

- formulate hypotheses regarding the function of the nervous system

The activity of formulating a hypothesis in terms of a model requires collection of experimental data. It is often discovered that crucial data are missing. In this respect a model could be considered

- a tool to identify what we don't know

Often, hidden contradictions and inconsistencies are revealed during the formulation process, and it happens that models don't yield expected results so that a further role of the model is

- validating self-consistency of the description of a phenomenon or function

If the confidence in the model is strong but the predictions differ from experiment the model can be used to

- falsify hypotheses

If phenomena in the model are unexpected or unobserved a model can

- suggest new experiments

Finally, a model can be used as a

- platform for integrating knowledge

unifying experimental data from many sources in a consistent manner.

4.2 Top-down and bottom-up approaches

The *top-down* approach to modeling most often means using hypotheses on the computational or algorithmic levels as a starting point when approaching the formulation of a model at the level of physical implementation. These functional hypotheses guide the formulation of the model at the implementational level in terms of what elements are included in the model and which experimental data are considered important. A functional hypothesis can also complement experimental data in the sense of giving additional constraints. For example, if we

have limited experimental data on the functional connectivity between inhibitory and excitatory connectivity, an additional functional constraint would be that the activity in the network must not be allowed to grow in an uncontrolled manner.

A top-down approach can also start with an abstract model, neglecting detail below the level of organization at which the model is formulated. Succeeding models can then increase the amount of detail, enabling them to account for a larger set of experimental data. For example, a connectionist, rate-based model can be developed into a spiking network model, followed by a Hodgkin-Huxley style model. However, it should be noted that models are more typically developed outside of such sequences, at a specific level of abstraction and detail suitable to the scientific questions posed.

The *bottom-up* approach, in contrast, means using the level of physical implementation as a starting point with the hope of capturing function. For example, what does the anatomy of the cerebral cortex mean? If we can, from the physical level of synapses, dendrites, neurons and networks, identify computational primitives of the cortex such primitives can be abstracted and we can move up one level of analysis. This is one goal of projects like DAISY (Kennedy, 2005), FACETS (Meier, 2005) and Blue Brain (Markram and Peck, 2004).

As with the top-down approach, a bottom-up approach can be concerned with levels of organization. In this case, it means to take a lower level of organization as a starting point for understanding the higher level.

In practise, the approach of a modeller is usually neither purely top-down nor purely bottom-up, as was already evident in Marr's work. Also, in the bottom-up approach, the focus of experiments and the choice of elements to include in the model is largely guided by functional hypotheses.

4.3 Explicitness

We define *model explicitness* to mean the degree to which the model is isomorphic with reality, or, how directly state variables of the model can be mapped to empirical data.

The degree of detail in a multi-compartment, Hodgkin-Huxley, model of a neuron aids in making this type of model transparent in the sense above. During intracellular recording of a neuron, it is often possible to block a subset of the ion channels in order to directly measure the current of another channel subset, or, in order to study the effect on cell behavior. The explicitness of the Hodgkin-Huxley formalism then makes it easy to perform a corresponding manipulation in the model. It is also easy to identify and display individual currents in the model.

In comparison, an integrate-and-fire model is less transparent. While recent versions (e.g. Brette and Gerstner) of this type of model can faithfully reproduce a spike train, some state variables correspond to the phenomenological effect of the coordinated action of multiple channel types in the real neuron. In this case, it is easy to compare the spike trains, but it is not as easy to map the dynamics of the model to the individual currents of the real neuron. On the other hand, the simplicity of the integrate-and-fire model makes it easier to analyze and understand.

Another aspect of explicitness is that it supports the role of the model as a platform for integrating knowledge. A transparent model is more likely to connect well to a wider range of experimental data, even data which were not targeted when constructing the model.

Workshop participants commented that, in the past, there have been many sacrifices in explicitness in order to achieve performance. Early connectionist models used simple rate-coding units, enabling the simulation of models of networks on the personal computers of the 1980s, while Hodgkin-Huxley type models could only be simulated one neuron at a time. Today, we can simulate large networks of neurons with thousands of compartments.

4.4 Large-scale models

For the purposes of this report, a *large-scale model* is a model with a high dimension, i.e. a model with a large number of state variables (on the order of hundreds of millions or more). Thus, a detailed model of one cortical column can be large-scale while an abstract model of a large network encompassing multiple columns is not necessarily large-scale.

4.4.1 Integrate-and-fire models

The classical integrate-and-fire model (MacGregor and Oliver, 1974; Tuckwell, 1988) has one state variable per neuron, representing the membrane potential. It is basically a linear leaky integrator with a voltage threshold and a reset mechanism. The main advantage of this type of model compared to Hodgkin-Huxley type models is its simplicity. For example, it has far fewer parameters, while it still captures essential features of the neurons. Because it has fewer parameters it is easier to adapt this type of model to experimental data. In this sense, it is easier to achieve a certain level of realism with an integrate-and-fire model than with a Hodgkin-Huxley type model, while, with more effort and more data, the latter model can reach an

even higher degree of realism. Its simplicity also makes it possible to develop automated procedures for extracting parameters from data (Jolivet et al., 2004; Keat et al., 2001; Paninski et al., 2004). Because of the low dimension and mathematical tractability, it is easier to analyze and understand this type of model. Finally, the simulation of this type of models require fewer computational resources making it possible to simulate larger networks given the same hardware. For examples of large integrate-and-fire-based network models, see Mehring et al. (2003); Aviel et al. (2003); Tetzlaff et al. (2004); Kumar et al. (2007); Morrison et al. (2007a).

The integrate-and-fire paradigm has recently been developed in three directions (Brette and Gerstner, 2005):

- the addition of a quadratic or exponential term, yielding a smooth spike initiation zone (Latham et al., 2000; Fourcaud-Trocme et al., 2003);
- the addition of a second state variable, enabling modeling of subthreshold resonances or adaptation (Izhikevich, 2003; Richardson et al., 2003);
- using active conductances to model synaptic inputs (Destexhe et al., 2003).

4.4.2 Hodgkin-Huxley models

A network model with multi-compartmental Hodgkin-Huxley type units is more detailed than its integrate-and-fire counterpart. It is more complex, both by having more parameters and a larger model dimension. As has been discussed in section 4.4.1, this requires more labor to determine parameters from experimental data. Sometimes, not all data is available so that parameters need to be determined more indirectly. The disadvantage is that this turns an output of the model into an input (3.2). For example, if we tune the conductance of a K_{Ca} channel in order to obtain the correct time course of an AHP, instead of measuring this conductance, we can no longer claim that our model predicts a correct AHP. In this case, however, the kind of predictions which are of interest in a network model appear at another level of organization within the model.

In section 4.3, the explicitness of HH-type models was discussed. The higher level of detail allows this type of model to be connected to a wider range of experimental data. The presence of ionic currents allows for comparatively easy modeling of pharmacological manipulations. The 3D extent of a compartmental model allows for the synthesis of EEG and LFP signals (Einevoll et al., 2007).

4.4.3 The bottom-up approach to the cortical column

Regardless of whether we use integrate-and-fire or Hodgkin-Huxley type units in a network model, an important set of parameters that currently largely lacks an experimental basis is the set of connectivity parameters. Data from, for example, Thomson et al. (2002) and Binzegger et al. (2004) gives the statistics of connectivity between pairs of cell types. This type of data has led to the use of random Gaussian (e.g. Brunel) or random uniform (e.g. Haeusler and Maass) connectivity in network models, consistent with such statistics. However, it is reasonable to assume that the microcircuitry of a column has more structure than that. Also, there is very limited data on the structure of long-range connectivity.

The Blue Brain project (Markram and Peck, 2004) aims to collect data on individual cells, for example acquisition of cell morphology through cell labeling and 2-photon microscopy, and then using database techniques and specialized software to reconstruct a virtual column. The superposition of reconstructed cells in 3D-space may give additional constraints needed to get a more complete picture of microcircuitry. The aim is also to simulate a large-scale network model of a complete column with multicompartmental Hodgkin-Huxley-type units without reference to functional hypotheses about the network. Thus, the approach is essentially hard-core bottom-up.

Workshop participants commented on the difficulty in experimentally determining the existence of a synaptic contact between cortical neurons since axonal processes can pass close to the dendritic processes of neurons without forming a synapse. In essence, the only way to determine anatomically if there is a contact is by looking at it with an electron microscope. It is also possible to determine the existence of a connection electrophysiologically by recording from pairs of neurons (e.g., Thomson et al.).

One particularly interesting development with regard to the acquisition of connectivity data was discussed at the workshop. Denk and Horstmann (2004) have developed a method called “serial block-face scanning electron microscopy” or SBFSEM. A microtome is placed in the chamber of a scanning electron microscope. The face of the tissue sample is scanned and 50–70 nm slices cut away, generating stacks of thousands of images from which a bulk 3D volume can be reconstructed. The acquired data has enough resolution to trace thin axons and identify synapses. This method holds the promise of geometrically reconstructing an entire neocortical minicolumn and extracting its circuitry.

Workshop participants concluded that the acquisition of data at multiple levels of organization leads to a new scale of projects—a development which parallels the study of the genome and studies in particle physics. There will be new kinds of challenges associated with industrial-scale data acquisition in projects of thousands of man-years.

4.4.4 Combining the top-down and bottom-up approaches

A survey of existing literature in computational neuroscience shows that the pure bottom-up approach to understanding network function is very rare. Part of the explanation is that missing empirical data need to be complemented with functional hypotheses in order to make progress possible. But a top-down approach may have importance in other ways than replacing lacking knowledge. During the workshop, the question arose whether a correctly implemented, detailed computer replica of the cortical column would by itself say very much about network function. Because a detailed model can be complex and hard to analyze and understand, modellers tend to see functional hypotheses and abstract models as a necessary complement in dissecting cortical function. It should still be noted that a computer replica is much more accessible to experimentation than the living tissue in the sense that any set of state variables can be logged and an arbitrary set of variables can be simultaneously perturbed in a precise fashion.

An example of how a top-down approach can be combined with the bottom-up approach is given by the model of Lundqvist et al. (2006) (see Djurfeldt et al. for a large-scale version of the model). The model is mainly designed to target the question: Is neocortical microarchitecture consistent with the hypothesis of attractor memory network function? Here, most parameters of the neuron models are determined from experimental data in a bottom-up manner. However, the connectivity parameters are determined by combining a long-range connectivity structure required for attractor memory network function with the currently existing empirical constraints on connectivity mentioned in section 4.4.3.

Another aspect of this model, and most or all other network models, is that the parameters of a neuron type are replicated over the population of model neurons, with or without random perturbation, in a crystal-like manner. This means that even if a large-scale model of this type has a large model dimension, it can still have a comparatively small number of parameters and, thus, be simple in the important sense (c.f. section 3.2).

4.4.5 Volume simulation

Until now, Hodgkin-Huxley type models have represented the most basic level of organization at which we simulate neurons and circuits, with the exception of hybrid models also including biochemical processes inside the cell. Data from the SBF-SEM method mentioned in section 4.4.3 opens up the prospect of a full 3D volume simulation of a cortical column. During the workshop, Gabriel Wittum presented initial attempts in this direction at the level of a single cell. In the Hodgkin-Huxley approach, the neuron is modelled as an electrical circuit. Here, instead, the 3D volume in which the neuron is embedded is described in its entirety by partial differential equations (PDEs) and simulated using the solver μG (Bastian et al., 1997; Wittum, 2007).

A model of the 3D volume can be based on *first principles* in the sense discussed in section 3.5. In this case, the model is based on Maxwell's equations which describe the dynamics of the electromagnetic field.

4.4.6 Simulation of growth processes

In order to fully understand the cortical architecture, it is necessary to understand the development and growth processes from which it results. Within the DAISY project, initial steps are currently taken to simulate the migration of neuroblasts. This adds a requirement on the simulator which is not yet fulfilled by standard neuron simulators such as Neuron and Genesis: there is a need to quantize space. Simulation tools are being developed within DAISY to meet this demand. The solver μG is based on a communication layer, DDD (Dynamic Distributed Data), which allows computational loads to migrate within a parallel computer during simulation. This layer could also be a suitable substrate for the simulation of growth processes.

4.5 Simulation versus emulation

Alan Turing used the term *simulation* in a very specialized sense. The term referred to the simulation by a digital computer of a subject discrete-state machine, defined by a set of state transitions, inputs and outputs.

In computational neuroscience, the term usually refers to the computation of the numerical approximation of a solution over time to the equations of a mathematical model. When performed on a digital computer, such computations are subject to the limitations of the computer. For example, some quantum mechanical processes can not be simulated on a digital computer. However, such limitations do not seem to be of any relevance to current computational models of the nervous system.

By an *emulation*, we refer to a model of the nervous system in the shape of a physical realization, for example in terms of an electronic circuit. The projects DAISY and FACETS both implement VLSI chips emulating neural circuits.

During the workshop, participants commented that while most simulations are slower than real-time (time simulated is shorter than the wall-clock time required to compute the solution), it is possible to construct an artefact which can emulate a neural circuit in real-time. The important consequence is that the artefact can interact with the environment in real time and react to a continuous flow of events occurring asynchronously. In some cases this is also achievable with a simulation on a digital computer, as exemplified by the dynamic-clamp technique (Sharp et al., 1993) or the goal of the Blue Brain project to simulate a cortical column on a super-computer in real time.

4.6 Upscaling

A model of the bulk 3D volume of neural tissue using PDEs (section 4.4.5), if well-rooted in empirical data, can be considered more realistic than the Hodgkin-Huxley model. It is based on first principles (sections 4.4.5, 3.5) while the Hodgkin-Huxley model is partly based on simplifying assumptions and curve fitting. This means that we can use the 3D volume model to *validate* the multicompartmental Hodgkin-Huxley model.

Gabriel Wittum reported on simulations of a single cell where ephaptic interactions could be observed between two of the dendritic processes of the cell itself. Such a phenomenon would not arise in the Hodgkin-Huxley model. It is also clear that the surrounding interstitial fluid, neuropil, and dendritic processes have substantial effects on the electric behavior of the cell membrane, diverging from the Hodgkin-Huxley model. Clearly, however, the 3D volume simulation is not a good replacement for the Hodgkin-Huxley model, because it is computationally heavier. The question then arises what alternatives we have to the HH model.

An important answer is given by the 3D model itself: There exist mathematical techniques for deriving a model at a coarser scale from a model at a finer scale. This methodology is called *upscaling* (Eberhard et al., 2004). Through upscaling techniques it might be possible to derive a candidate model which might serve as a better replacement for the Hodgkin-Huxley model.

5 Software for large-scale simulations

5.1 Important properties of simulator software

When judging software for large-scale simulation, there are many criteria that need to be examined, some of which will be mentioned in this section. Brette et al. (2007) reviews existing tools for simulation of networks of spiking neurons.

- **Model types supported** What neuronal/synaptic/plasticity models can be simulated?
- **Accuracy** Does the simulator give correct results? Recent work (Brette, 2006; Rudolph and Destexhe, 2006; Morrison et al., 2007b) has presented methods for determining spike times in a simulator more precisely, and has shown that this can have effects on discharge statistics and temporal precision in resolving synaptic inputs. This is discussed further in section 5.4 below.
- **Scalability** In general, it is difficult to write efficient parallel implementations. How does *speedup* (simulation time divided by simulation time on one processor) scale with the number of processors used? Ideally it should grow linearly. How does simulation time scale as a function of the size of the model?
- **Documentation** How good is the documentation?
- **Support** How quickly will the developers respond to bug-reports or feature requests?
- **License and availability of source code** In a research environment, it is an advantage to have the source code for the simulator available, and to have permission to modify it. This is guaranteed for all software covered by the GPL license from the Free Software Foundation and some related licenses.
- **Adaptability** How easy is it to adapt the simulator to your purpose? How easy is to to add new mechanisms?
- **Portability** Does it run on my preferred platform?
- **Interoperability** How easy is it to collaborate with others using a different simulator? This is discussed further in section 5.3 below.
- Is there a graphical interface?
- What analysis/post-processing tools are available?

5.2 Diversity of simulators

During the workshop, it was discussed whether it is sensible for the community to split efforts into the large and growing set of neuron simulators available today rather than focussing on one or a few tools. However, it was also noted that there is currently a strong ongoing development of simulation technology, and that simulators tend to have unique strong points not shared by others.[1]

The workshop also identified the reproducibility of models as a major problem (see section 6.4). The diversity of simulators might allow for simulating the model on a different platform from that on which it was originally developed, thereby verifying the reproducibility of results. This leads to the first major finding:

Workshop participants agreed that the current diversity of simulators creates vigor in the field and has benefits for the validation of models.

5.3 Software interoperability

Given the diversity of simulators, it is important to find ways to share the gains produced by the efforts, of both developers and users, put into individual simulators. During the workshop some approaches to simulator-independent modeling environments were discussed, which allow a model to be simulated using more than one simulation engine. This is important for verification of simulator accuracy, for the reproduction, testing and extension of published models, and for collaboration between modellers using different simulation tools. The approaches discussed were graphical environments, declarative model specifications using XML, and procedural model specifications using an API implemented in the Python programming language (see sections 5.3.4 and 5.6). Another conclusion was that more effort should be put into the development of such simulator-independent environments ('meta-simulators'). Thus, the second major finding of the workshop was:

Workshop participants agree upon the importance of facilitating software interoperability and re-use of simulation software components.

5.3.1 Modularity

The practise of dividing software into *modules* with well-defined roles has many advantages. It eases development and increases maintainability of the code. If such modules have well-defined *interfaces*, modules can be re-used in other circumstances. Such an interface can be in the form of

1. an *application programming interface* (API), enabling a module in the form of a compilation unit or a library to be linked into an application. This includes the definition of *data structures* required to pass information through the interface.

2. a *communication interface*, enabling modules in the form of processes to communicate while running simultaneously on the same or on different machines

3. a *file format*, allowing the output of a module in the form of an application to be read as input to another application. Here, "input" can be a model specification. In this case, the interface takes the shape of a model specification language.

As an example of the first form of modularity, a simulator can be divided into a *simulator kernel*, responsible for the distribution and allocation of data structures over a cluster, for building the model on the nodes, and for performing the computations during a simulation, and other modules required for module specification, input and output. The simulator kernel can be further divided into a *solver*, with the sole responsibility of performing computations, and modules required for allocation, distribution etc.

Workshop participants discussed the possibility of on-line interaction between simulators (see section 5.3.2). This would entail modularity of the second type.

An example of the third type of modularity is neuroConstruct which is a software application for creating 3D models of networks of biologically realistic neurons through a graphical user interface (GUI) (7.1.3). neuroConstruct can import morphology files in Genesis, Neuron, Neurolucida, SWC and MorphML formats for inclusion in network models and can generate model specification files for Genesis and Neuron. Efforts put into developing neuroConstruct further will thus benefit both the Genesis and Neuron communities. Note, though, that the choice of two output formats is forced by the current lack of a standard format for model description. This will be discussed further in section 5.3.3.

David Beeman presented how the next generation of Genesis, Genesis 3 (7.2.2), aims to foster collaborative modeling through a rich set of interfaces.

1 Michael Hines noted that "The reason why we keep reinventing the wheel is that we haven't got it quite round yet."

We cannot gain fully from a modularization of simulation software until we have developed *standard interfaces* between software components.

When facing the need to develop standards, there is the possibility of assigning the task to a standardization committee. Workshop participants agreed, however, that there are other alternatives which might be preferable. It was suggested that a workshop should be arranged with development of interoperability as one goal.

Participants reported on good experiences with the form of the Telluride workshops (van Schaik et al., 2006) which includes background lectures from leading researchers, practical tutorials from state-of-the-art practitioners, hands-on projects involving established researchers and newcomers/students, and special interest discussion groups proposed by the workshop participants. The INCF workshop gives the following recommendation:

Arrange an annual workshop on large-scale modeling with the aim of developing standards for improving reusability of software. A workshop in the format of the Telluride Workshops on neuromorphic engineering would give an opportunity to teach large-scale simulation technology while simultaneously presenting an optimal environment for practitioners and student to discuss, test, and evaluate approaches to software interoperability.

5.3.2 On-line interaction between simulators

As was discussed in section 5.2, different simulators have different strengths. Workshop participants observed that current development is moving in the direction of simulation of large systems of networks. The situation could arise that one simulation framework is most suitable for one part of the system while another framework is needed for another part. For example, a retina model could provide input for a model of LGN/PGN/V1, or, a medium-sized network of detailed structurally realistic neurons could interact with a very large network of simple integrate-and-fire point neurons. In the former example, one solution would be to perform the simulation in *batch mode*, letting the retina model generate spikes and save the spike times to file. Such spike files can then be read by the second simulator. This would be an example of the third type of modularity in section 5.3.1. In the latter example, this is not possible due to the need for bi-directional interaction. This situation is resolved if the simulators can transfer spikes *on-line* through a communication interface.

Even in cases where batch mode simulation is possible, it may be desirable to let such a simulation interact with the environment in real time. Another argument for preferring on-line communication between simulators over batch mode simulation is if the amount of data generated is of such a large magnitude that it is undesirable or impossible to store intermediate data in files.

If both simulators are run on the same parallel computer, some further coordination may be required with regard to allocation of nodes, initialization of MPI, etc. This could be achieved if the communication interface has the form of a communications library, providing services like initialization of MPI and the option of external communication. There is also a need for a naming or addressing mechanism to allow for flexible connectivity between modules through multiple communication links.

The development of such an interface may or may not result in a software component that will be generally adopted, but has value in itself as an exploration of the concept. The workshop gives the following recommendation:

Implement an experimental framework for connecting software components. A feasibility study should be performed regarding the possibility of on-line communication between different software modules, for example two parallel simulators. INCF should allocate resources for implementing a software library with a communication interface.

5.3.3 Common specification language

Different simulator environments have different ways of specifying a model. For example, Neuron and Genesis have distinct specification languages. The translation of a model description from one environment to another can entail substantial effort. This is especially true in the case that environments use different formalisms. If the differential equation for the activation factor in the Hodgkin-Huxley model of a channel has different forms in the two environments, the translation of the model will even involve finding a new set of parameters. This situation creates barriers between laboratories using different simulators, makes it harder for one laboratory to freely choose the tool suitable for the problem at hand, and threatens the reproducibility of scientific results. A common, standard specification language, supported by all simulators, would alleviate such problems and increase the utility of model repositories such as ModelDB (7.5.1) and databases such as NeuronDB (7.5.2).

NeuroML (7.3.1) was discussed as one viable candidate for a model description standard. NeuroML is a collection of projects with the aim of developing standards for specification of neuroscience models in XML. It is organized into four *levels of scale*, where each succeeding level extends the features of the language. The first level covers the neuronal level of organization while the last level (level 4) covers all levels of organization from biochemical networks to systems. It includes the XML schemas MorphML, ChannelML and NetworkML. Workshop participants noted that NetworkML is currently the least developed schema and needs input from simulator developers.

PyNN, discussed in section 5.3.4, was also proposed as a candidate for a standard model description.

5.3.4 Common scripting language

A *scripting language* is an interpreted language which is used to control an application. It is often *embedded* in the sense that much of the functionality of the application is available as function or procedure calls in the language—a *language binding*. If it is possible to add new functionality to the application in terms of code written in the scripting language, it is also called an *extension language*. The use of a full-fledged general purpose programming language as scripting/extension language has emerged as a powerful concept. Scripting languages such as Emacs Lisp, TCL, Matlab and Python have each given rise to prolific user communities and rich sets of software tools and libraries, and can provide a backbone in a framework with multiple modules.

Within the FACETS project (Meier, 2005), Andrew Davison has proposed the PyNN framework (7.3.2) as a standard scripting language binding for neuronal network simulators. This abstracts differences between simulators and provides a common way to specify models and run simulations. The goal is that simulation scripts in PyNN for simulator A will run on simulator B without modification.

PyNN is based on the Python programming language and includes the development of an API and the binding to individual simulation engines. The API has two parts, a low-level, procedural API, and a high-level, object-oriented API. The low-level API can be used for small networks, and gives more flexibility than the high-level API. The high-level API hides details and book-keeping, and is intended to have a one-to-one mapping with NeuroML, i.e. a population element in NeuroML will correspond to a Population object in PyNN, etc. Another requirement for a common scripting language is standard cell models. PyNN translates standard cell-model names and parameter names into simulator-specific names.

The use of Python in PyNN results in a free, Matlab-like environment with tools for data analysis, plotting, mathematical libraries, etc., leveraging the efforts of the Python user community.

5.4 Accuracy of simulation

5.4.1 Combining off-grid spike events with time-driven global scheduling

Very large networks of spiking neurons can be simulated efficiently in parallel under the constraint that spike times are bound to an equidistant time grid. Within this scheme, the subthreshold dynamics of a wide class of integrate-and-fire type neuron models can be integrated exactly from one grid point to the next. However, the loss in accuracy caused by restricting spike times to the grid can have undesirable consequences, which has led to interest in interpolating spike times between the grid points to retrieve an adequate representation of network dynamics. Markus Diesmann demonstrated during the workshop how the exact integration scheme can be combined naturally with off-grid spike events found by interpolation (Morrison et al., 2007b). By exploiting the existence of a minimal synaptic propagation delay, the need for a central event queue is removed, so that the precision of event-driven simulation on the level of single neurons is combined with the efficiency of time-driven global scheduling. Further, for neuron models with linear subthreshold dynamics, even local event queuing can be avoided, resulting in much greater efficiency on the single neuron level. A measure of the efficiency of network simulations in terms of their integration error shows that, for a wide range of input spike rates, these novel techniques are both more accurate and faster than standard techniques.

5.5 Preprocessing and specification of large-scale network models

Section 4.4.3 discussed the bottom-up approach to large-scale modeling of the cortical column using data from the SBFSEM method. For a 3D volume simulation (section 4.4.5), a corresponding 3D volume of dielectric properties must be generated as part of the model specification. For a more traditional simulation based on compartment model neurons, the data must be analyzed with respect to cell morphology and a list of network connections be generated.

Section 4.4.4 discussed a combined top-down/bottom-up approach where the connectivity is based on functional hypotheses regarding cortical function. The Brunel (2000) and Haeusler and Maass (2007) models also belong to this type. In such models, there is a large contrast between the seemingly

complex connectivity of the simulated model and the simple ideas upon which the connectivity is based. Usually the connectivity is expressed by a serial algorithm that sets up connections one by one. On a parallel machine, several simulators use the approach of running the serial algorithm in each process independently and simply ignoring attempts to set up connections not belonging to the local process. The advantage is that the complexity of parallelism is hidden from the user. However, this approach does not scale well. Djurfeldt et al. (2005) presents an approach which scales well at the same time as expressing the connectivity in a form that preserves the underlying ideas explicitly in the program code. Projections between neuronal populations are described by iterators representing infinite connection matrices. Only the relevant finite piece of a matrix is used to connect the populations during model setup. A basic set of parameterized matrices can be combined through a connection matrix algebra to form new connectivity structures.

A third way to specify networks is to generate them based on empirical structure parameters. Regarding automatic generation of realistic, large-scale neural networks, there is a need to distinguish between generation and growing of cells. Growing of cells might depend on local or global parameters in the network. Further, an open question is how to specify and generate spatial connectivity. The easiest way of generating connectivity is using experimental data about the localization of different neuron types in the brain. This approach is commonly used in the available software tools. During the workshop, Jens Eberhard presented an assessment of existing tools and evaluated features such as interfaces and used formats, "large-scale" capability, expandability, and postprocessing/visualization. The following tools were evaluated: L-Neuron (7.1.1), ArborVitae (7.1.2), A-Cell-3D, NeuroConstruct (7.1.3), and NeuGen (7.1.4). Based on the features mentioned above, only NeuroConstruct and NeuGen could be recommended for the specification of large-scale network models. These tools are capable of generating detailed three-dimensional neuron geometry for networks up to about 10000 cells.

5.6 Declarative versus procedural model description

One of the discussions during the workshop concerned the relative merits of declarative versus procedural model specification. Model definitions based on NeuroML are basically declarative, although it is in principle possible to specify algorithmic elements in XML. In contrast, a model description in a Python-based scripting language is often procedural.

A declarative description is often easier to read and understand than a procedural description. It also usually gives a software tool more leeway for internal optimizations.

However, workshop participants commented that, in a certain respect, a declarative description is "un-human". This is because it throws away structural knowledge about the problem already available to the researcher. This knowledge is expressed in algorithms which can be close to the mathematical formulation of the model. We should look for a good high-level description that allows a human reader to see from the code what the point is.

5.7 Postprocessing and visualization

The kind of large-scale simulations which are the subject of this workshop have, by definition, a large number of state variables. When the number of variables is in the order of hundreds of millions or more, the problem how to visualize network activity becomes important. One of the tools discussed during the workshop was DAVIS (7.4.1). However, there seems to be a great need of further tool development in this area. For example, for sufficiently large data volumes, the visualization tool needs to be parallelized.

A large-scale model spans many levels of organization, and, thus, needs to connect to experimental data on many levels. There is, therefore, a need for tools which can transform logged data from model state variables, such as cell membrane voltages, into synthetic versions of common brain imaging methods such as fMRI, VSD and EEG.

6 Infrastructure

6.1 The role of the INCF with regard to large-scale modeling

There was a consensus among workshop participants that the field of large-scale modeling of the nervous system has reached a level of maturity where it would benefit from coordination efforts, for example with regard to simulation tools, databases and publication of results. Various specific ways in which INCF could assist such efforts were discussed, some of which are summarized in the following sections.

One of the roles suggested for the INCF was to develop an expert definition of where things stand in the field and identify complementarity between research groups, i.e. make an inventory of what is available in different places, define areas in which it is important to achieve progress, and facilitate this. The workshop gives the following recommendation:

Continue work on defining areas in need of support and coordination. Workshop participants agreed that the 1st INCF Workshop on Large-scale Modeling of the Nervous System was successful. However, there is still a great need for further work on coordination within the field. INCF should therefore arrange further workshops with the aim of defining areas in need of support and coordination.

It was also suggested that the INCF could have a staff of experts available who understand various software applications and components fully, and know how to run tools on a reference set of problems. This could provide a resource both with regard to development of software interoperability (c.f. section 5.3) and teaching.

With regard to databases and platforms (see section 6.6), it was suggested that the INCF should promote and coordinate data archives. What data are there? What is missing?

Workshop participants reported on a general feeling in the modeling community that we need and can achieve synergy effects if money were to be allocated for people to guide the development of software interoperability and common platforms. The comment was also made that administrators of funding agencies would surely recognize this need if it was properly communicated to them. It was concluded that there is a communication problem between researchers and funding agencies in this respect and it was suggested than the INCF could aid in this communication, for example vis-a-vis the European Union. More generally, it was suggested that the INCF could assist in networking people in the field.

6.2 Verification of simulator function

In science, an individual experiment carries little weight, but when the result is reproduced in other laboratories, by different people, in slightly different ways, this gives strong validation. One of the major discussion topics during the workshop was a parallel need for validation of simulator function.

It is an established software engineering fact that any sufficiently complex program does contain errors, regardless of the quality of the developer team. For a simulator, there can be errors at the level of a model built in implicitly in a simulator—e.g. the Hodgkin-Huxley neuron model, at the level of the numerical method used, and at the level of its implementation. Furthermore, different simulators can have different accuracy and vary in their efficiency.

Section 5.2 discusses the diverse set of existing neuron simulators. One of the ways in which to draw advantage from this diversity is to use the fact that different simulators are programmed differently to cross-validate simulators: If simulation results can be reproduced using a differently programmed simulator, this gives a relatively strong verification of the correctness of both simulators.

For such cross-validation to work well, there is a need to agree on a common set of simulator "benchmarks". Such a framework could also make possible comparisons of accuracy and efficiency. This could work as an inspiration and driving force for simulator developers, and help in focussing development. Without quantitative comparison data it is difficult for developers to know what is practically feasible.

If a benchmark suite is published on web pages maintained by the INCF this would be of great community value. It is important that such benchmarks be with respect to published models not contrived by the developers. Workshop participants mentioned cases where essentially good methods were neglected because they had poor performance in artificial benchmarks. This experience parallels that in software engineering, where the trend during the previous two decades has been from artificial contrived benchmarks towards more application-like. It is also important that the data on the web pages be updateable so that old information can be superseded by current best practices in each simulator.

Examples of possible benchmarks are the DeSchutter Purkinje cell model (Schutter, 1998) and the model of Vogels and Abbott (2005). ModelDB (7.5.1) contains 280+ such published models from which a selection could be drawn. The workshop gives the following recommendation:

Implement a standard simulator test suite. 1. An INCF staff member should propose a set of simulator "benchmarks" for the purposes of verifying correctness of computation and serving as a standard for simulator performance. These should preferably be selected with respect to published models not contrived by the simulator developers. 2. INCF should allocate funding for and coordinate the implementation of the test suite by members of the labs that have developed the simulators. 3. INCF should maintain web pages with the results of the test suite for each simulator. These pages should be updateable so that old information can be superseded by current best practices in each simulator.

6.3 Model verification

In section 6.2, the verification of the simulation tools was discussed. During the workshop there was also a discussion about the verification of *models* and the reproducibility of modeling results.

A crucial point for a simulation is: How do you know that what you're simulating is correct? As was discussed in section 3.5, the model needs to be well rooted in empirical data. This can be achieved if test problems are chosen with care. Such problems should deal with basic traits of the domain under study and be chosen so that many aspects of the problem are easily accessible through experimental measurement. The test problems can validate the simulation technique so that simulations can then be extrapolated to other problems with some degree of confidence. The comment was made that some areas in the brain are easier to work with, with regard to model validation, than others. It is especially important to look at a region with well-defined inputs and outputs, such as, for example, the barrel cortex.

Section 3.5 also discussed how models based on first principles, such as a 3D volume model of the electromagnetic field in neural tissue, increase confidence in the model; section 4.6 discussed how this gives the possibility of validating the Hodgkin-Huxley model and how upscaling techniques could be used to derive, from the 3D volume model, a new model at the same level of organization as the Hodgkin-Huxley model.

From the top-down perspective, doubt was raised whether a 3D volume model of an entire network, such as those discussed in section 4.4.3, would really say anything, and, in particular, that such very detailed network models lack measures of verifiability. The approach suggested to the latter problem was to verify the simulation tool on simpler problems, such as the test problems discussed above.

In order to develop good test problems, there is a need for experiments tailored towards simulation. It was pointed out that there is a sociological, or structural problem in the field such that experimentalists will not go very far in answering such needs. On the other hand, there are now some experimental laboratories which use modeling as a fundamental research tool.

6.4 Model reproducibility

There was a consensus that published simulation results are, in general, very hard to reproduce. Thus, the third major finding of the workshop was:

Workshop participants reported on difficulties in reproducing simulations from published articles.

There seem to be several common reasons for this:

- The model description is incomplete. It is often necessary to contact the authors to get additional information required to run the simulation.
- The model description differs from the model used to produce the results. It occurs that last-minute modifications are made to the model, while the same modifications do not find their way into the methods section.
- The model description uses a formalism not supported on the simulator used to reproduce the results. One example of this is given in section 5.3.3.

It may also happen that the simulations described in the article have been carried out on peculiar hardware and/or with custom code, entailing substantial re-implementation efforts when running the simulation with standard software on standard hardware.

Since reproducibility is one of the tenets of science, the workshop gives the following recommendation:

Develop publication guidelines for ensuring the reproducibility of simulations. INCF should develop guidelines for publications with computer simulations and publish these on the INCF homepage as a reference for authors. INCF should also encourage publishers to implement policies for making models available in conjunction with publication.

One topic discussed at the workshop was whether the concept of reproducibility should include exact quantitative reproducibility, i.e. reproduction of not only the behavior of the model

but also the exact numerical values of simulation output. Such a requirement is harsh, because it means not only that identical algorithms must be used, but also that numerical operations need to be performed in an identical order so that cancellation effects, rounding errors, etc. do not cause noticeable differences.

An intermediate requirement would be that every simulation of the model with the same simulation software on the same machine gives identical results. This kind of reproducibility is important, for example during development and debugging. Even this is far from trivial on a cluster, since this requires a mechanism for seeding the pseudo random number generator in each process. It may even be impossible to make the result independent of the number of processes, because the uses of random numbers occur in a different order depending on how the model is distributed over processes.

6.5 Method development in computational neuroscience

From a developers point of view, it was pointed out that the strong commitment required to develop an application is not sufficiently recognized. This is especially the case when making the transition from research tool to public tool, because this requires documentation, user support and high code quality, at least in the case of an open source project. Workshop participants reported on the difficulty in persuading funding agencies to fund simulator development. A quick round through the simulator developers present at the workshop showed that money supporting the development consistently came from a project *not* having simulator development as its main goal.

Large-scale simulation on parallel computers brings a new degree of complexity into simulation methodology which increases the need also for more theoretical method development. Again, it seems as if such research falls between the chairs with regard to funding—it is neither pure computer science nor neuroscience. The workshop therefore gives the following recommendation:

Encourage, support and fund work on method development within computational neuroscience. INCF should inform funding agencies about the need for research on methods in computational neuroscience, and on methods for large-scale simulation in particular.

In a similar vein, workshop participants reported on the difficulty in publishing simulation methods in peer-reviewed journals. One of the first peer reviewed journal articles on large-scale simulation is Morrison et al. (2005). The workshop gives the following recommendation:

Encourage, support and fund the production of publications on concepts and techniques for large-scale simulations. INCF should inform publishers about the need for published work on methods within the field of large-scale simulations and computational neuroscience in general, and about the current growth in this area.

6.6 A cyber infrastructure for computational neuroscience?

Workshop participants recognized the need to collect models into model repositories. This is important, not only for simulator verification and benchmarking (section 6.2) but also to aid development of large-scale network models. For example, there is a need for a library of cell models to draw upon when building network models.

Examples of existing model repositories are ModelDB (7.5.1) and the RIKEN J-Node portal/server (section 6.6.1). Models in these repositories can be coded in any specification language. The only requirement is to supply a README file. In the future, it may be desirable to develop model repositories conforming to a standard model description language (c.f. section 5.3.3).

6.6.1 The NRV project

The NRV project (Neuroinformation Research in Vision), at the Japanese neuroinformatics node (http://www.neuroinf.jp/), represents a mathematical modeling approach for understanding the brain as a system through the integration of experimental data at multiple levels.

The current paradigm for information flow in science today involves the use of many types of information technology, but what is finally stored on the PC is scientific papers. The NRV project aims to extend this paradigm by providing databases with peer reviewed content which can be shared in a manner similar to the sharing of papers. These databases contain:

- source code of models
- experimental data
- documents

The *Visiome platform* (Usui (2003); Usui et al. (2004); 7.5.4) is a first example of such a database. Concepts from Visiome have been used in development of the *XooNIps content man-*

agement system (7.5.5), which can be used as a base platform for many types of database from small to large. XooNIps is open-source software.

Other platforms at the Japanese neuroinformatics node include:

- The Neuron/Glia Platform, which integrates accumulating knowledge on the complex interactions between neuron and glia, including the activities of associated functional proteins.

- The Brain Machine Interface Platform (BMI PF), which accumulates BMI (Brain Machine Interface)-related experimental data, mathematical models, and tools generated in neuroscience, computational theory, and robotics.

- The Invertebrate Brain Platform (IVB PF), which integrates experimental data, mathematical models, and research tools relevant to the study of invertebrate brains, neurons, and behavior.

- The Cerebellar Development Transcriptome Database (CDT-DB), which provides spatio-temporal gene expression profile information on the postnatal development of mouse cerebellum. It seeks to reproduce the genetic blueprint of cerebellar development.

Platforms for neuroimaging and integrative brain research are currently under development.

Database platforms are integrated through the RIKEN J-Node portal/server where the XooNIps system is the key feature for inter-database communication.

The NRV project has as one goal to prepare for the arrival of a 10 PFLOP next-generation supercomputer expected to be installed at RIKEN in 2011.

It was observed by workshop participants that legislation in some countries, for example Norway and Sweden, requires the storage of research data for 10 years. It was suggested that this kind of tool might be suitable for this task and such use would entail many bonus benefits.

Another observation is that it would be motivating for individual researchers to contribute if publishing data in this kind of digital database would entail an increased likelihood of funding.

6.6.2 GRID

GRID technology is a method of pooling computing resources, such as computing, storage and network resources, so that they can be shared on a demand basis. GRID computing was suggested at the workshop as a possibility for sharing cluster resources for large-scale simulations.

Markus Diesmann informed workshop participants about the Black Forest Grid (Backofen et al. (2006); 7.6.1), which is a collaborative effort of ten faculties of the Albert-Ludwigs University to establish joint grid computing facilities using the framework and experience of the largest computing grid in the world, the LHC (Large Hadron Collider) Computing Grid. Such a framework is highly scalable and the software involved has been successfully used at CERN to manage 16000 contributed CPUs and 4000 TB of storage and to operate clusters of more than 2400 nodes. Using the grid infrastructure, the BFG has founded the virtual organization (VO) *CNS* as a starting point to explore the sharing of resources in Computational Neuroscience (Potjans et al., 2007).

Today, the GRID is not widely used for parallel processing requiring low latency communication between the compute nodes. However, this is just the constraint imposed by large-scale simulations of neuronal networks. Thus, from the Computational Neuroscience point of view, the grid would be mainly used as a grid of high-performance clusters. Here, the parallel simulation jobs are distributed across the grid, but each job runs at a single site. Whether the global and local scheduling mechanisms are already suitable for this type of usage needs to be explored.

Appendix A: Summary of existing and planned tools relevant to large-scale modeling

A.1 Model construction

A.1.1 L-Neuron

L-Neuron is a modeling tool for generating anatomically accurate neuronal analogues based on Lindenmayer systems. It is oriented towards single cell generation.

http://krasnow.gmu.edu/L-Neuron/

A.1.2 ArborVitae

ArborVitae (Senft and Ascoli, 1999) is software for the reconstruction of networks by algorithmic amplification of morphological data. It can be used to generate large-scale, anatomically accurate networks

A.1.3 Neuroconstruct

neuroConstruct has been designed to simplify development of complex networks of biologically realistic neurons, i.e. models incorporating dendritic morphologies and realistic cell membrane conductances. It is implemented in Java and generates output for the NEURON and GENESIS simulators. It uses the latest XML-based specifications for neuronal information exchange, including MorphML and ChannelML.

http://www.neuroconstruct.org/

A.1.4 NeuGen

NeuGen can generate realistic dendritic and axonal morphology of neurons and networks in 3D and construct networks of synaptically connected neurons in a cortical column. It uses experimental data to extract anatomical "fingerprints" of cells for synthetic generation.

http://neugen.uni-hd.de/

A.2 Simulators and environments

A.2.1 Parallel Neuron

The present standard distribution of NEURON, version 5.9, supports parallel simulation of network models in which cells on different processors are coupled by discrete logical spike events.

http://www.neuron.yale.edu/neuron/docs/help/neuron/neuron/classes/parcon.html#ParallelNetwork

A.2.2 Genesis 3

The GENESIS simulator is undergoing a major redevelopment effort that will result in GENESIS version 3. Genesis 3 has a modular architecture based on plugins and has multiple interfaces. This allows the use of multiple simulation engines, such as modules from MOOSE or Neurospaces, to perform the numerical calculations of the simulation. Models can be specified in NeuroML, making it possible to easily exchange models between simulators that use this standard. Advanced Java-based network visualization programs can be used interactively with GENESIS simulations. GENESIS 3 can be directly interfaced to other simulators, in order to allow simulations to interact. Simulations can be run from other modeling environments, such as the Systems Biology Workbench or the Modeler's Workspace. A WWW interface can allow remote simulations to be run, using a web browser. Alternate parsers can replace the GENESIS Script Language Interpreter to allow the use of improved or more standard scripting languages, such as Python. Estimated current development status is 1 yr before modeling is possible for general users.

http://www.genesis-sim.org/GENESIS
http://moose-g3.sourceforge.net
http://neurospaces.sourceforge.net

A.2.3 NEST

NEST is a simulation system for large networks of biologically realistic (spiking) neurons. It is best suited for the simulation of large networks of spiking point-neuron models where a realistic number of synapses need to be represented. The internal dynamics of these models may be arbitrarily complex (or simple). There is software support for heterogeneity in the neuron types as well as in synaptic dynamics including short-term plasticity and STDP. Neurons can interact by spikes constrained to the simulation time grid or in continuous time. NEST uses a hybrid multi-threading and message-passing simulation scheme. For more details see the corresponding section in Brette et al. (2007) as well as the following URLs.

http://www.scholarpedia.org/article/NEST
http://www.nest-initiative.org

A.2.4 NCS

NCS is a brain simulator which is optimized to model the horizontally dispersed, vertically layered distribution of neurons characteristic of the mammalian neocortex. The largest simulations to-date have been on the order of a million single-compartment neurons and 1 trillion synapses.

http://brain.unr.edu/ncsDocs/

A.2.5 CSIM and PCSIM

CSIM efficiently simulates medium sized networks of integrate-and-fire neurons. It is extendable and has an interface to Matlab. PCSIM provides a powerful environments for setting up and simulating large-scale networks, performing data analysis and visualizing results. In PCSIM the Matlab interface has been replaced with a Python interface.

http://www.lsm.tugraz.at/csim/
http://sourceforge.net/projects/pcsim

A.2.6 LENS

LENS is a simulation environment designed to support and integrate a wide range of brain modeling tasks across a variety of computational platforms. In contrast to most neural modeling environments, LENS supports arbitrary models and levels of abstraction, from neural compartments, neurons, and microcircuits to brain structures and global brain processes. LENS simulations are configured and initialized using a dynamically extensible specification language in neuroscience domain concepts and terms, which insulates model developers from non-model-specific complexity. LENS is developed at Biometaphorical Computing Research, IBM T.J. Watson Research Center, Yorktown Heights, N.Y., U.S.A. For further information, contact Charles Peck <cpeck@us.ibm.com>.

A.2.7 SPLIT

The SPLIT simulator (Hammarlund and Ekeberg, 1998) was developed in the mid 90's with the aim of exploring how to efficiently use the resources of various parallel computer architectures for neural simulations. It is a C++ library which is linked into the user program, which is serial, and can be linked with a serial or parallel version of the library. Parallelism is thus completely hidden from the user. SPLIT was recently optimized for the Blue Gene/L computer (Djurfeldt et al., 2005) and shows linear scaling behavior up to at least 4096 processors. The largest simulation performed with SPLIT comprised 22 million neurons and 11 billion synapses. For further information, contact Anders Lansner <ala@csc.kth.se>.

A.2.8 Systems Biology Workbench

The Systems Biology Workbench (SBW), is a software framework that allows heterogeneous application components—written in diverse programming languages and running on different platforms—to communicate and use each others' capabilities via a fast binary encoded-message system.

http://sbw.kgi.edu/research/sbwIntro.htm

A.3 Languages and language tools

A.3.1 NeuroML

NeuroML is an ongoing collaborative effort to support the use of XML for software relating to neuroscience modeling. It is not a single standard XML language - but rather a collection of related XML projects for modeling different aspects and levels of neural systems, from intracellular mechanisms and ion channel kinetics to the dynamics of networks of reconstructed neurons.

http://www.neuroml.org/

A.3.2 PyNN

The PyNN framework has been developed within the FACETS project (Meier, 2005) as a standard scripting language binding for neuronal network simulators. This abstracts differences between simulators and provides a common way to specify models and run simulations. The goal is that simulation scripts in PyNN for simulator A will run on simulator B without modification. PyNN is based on the Python programming language. The API has two parts, a low-level, procedural API, and a high-level, object-oriented API which is intended to have a one-to-one mapping with NeuroML.

http://pynn.gforge.inria.fr/

A.4 Visualization

A.4.1 DAVIS

DAVIS is a visualization system for cortical data and models.

http://vip.cs.utsa.edu/research/Davis/

A.5 Databases and Database tools

A.5.1 ModelDB

ModelDB provides an accessible location for storing and efficiently retrieving computational neuroscience models. ModelDB is tightly coupled with NeuronDB (7.5.2). Models can be coded in any language for any environment. Model code can be viewed before downloading and browsers can be set to auto-launch the models.

http://senselab.med.yale.edu/senselab/modeldb/

A.5.2 NeuronDB

NeuronDB provides a dynamically searchable database of three types of neuronal properties: voltage gated conductances, neurotransmitter receptors, and neurotransmitter substances. It contains tools that provide for integration of these properties in a given type of neuron and compartment, and for comparison of properties across different types of neurons and compartments.

http://senselab.med.yale.edu/senselab/NeuronDB/

A.5.3 BioModels

http://www.ebi.ac.uk/biomodels/

A.5.4 Visiome

Visiome Platform is a digital research resource archive for vision science. The available resources include mathematical models, experimental stimuli, experimental data, and analytical tools.

http://platform.visiome.neuroinf.jp/

A.5.5 XooNIps

XooNIps is an extension of the content management system XOOPS for the needs of neuroinformatics. The XooNIps-based database stores digital resources such as research articles, experimental data, mathematical models, and stimulations. It is a modular base platform which can support many types of database from small to large.

http://xoonips.sourceforge.jp/

A.6 GRID computing

A.6.1 Black Forest Grid initiative

The BFG is a collaborative effort of ten faculties of the Albert-Ludwigs University to establish joint grid computing facilities using the framework and experience of the largest computing grid in the world, the LHC (Large Hadron Collider) Computing Grid, LHC.

http://www.bfg.uni-freiburg.de

Appendix B: Workshop program

December 12:

12.00 – 19.00	Scientific presentations and discussions
Jan Bjaalie	About INCF
Anders Lansner	Topics and objective of the workshop
Rodney Douglas	Vision and demands for large-scale network simulations
Shiro Usui	Japan Node: supporting mathematical modeling for understanding brain function
Andrew Davison	Overview of simulators
David Beeman	Parallel simulation with GENESIS-PGENESIS
Michael Hines	Parallel simulation with Neuron
Markus Diesmann	Continuous spike times and efficiency in parallel simulations with NEST
Fred Harris	Parallel simulation with NCS
Mikael Djurfeldt	Parallel simulation with SPLIT

December 13:

08.30 – 13.00	Scientific presentations and discussions, draft report
Charles Peck	Parallel simulation with LENS
Thomas Natschäger	Parallel simulation with CSIM
Jens Eberhard	Specifying networks, growing neurons
Gabriel Wittum	Numerics of parallel neural simulations

References

Aviel, Y., Mehring, C., Abeles, M., and Horn, D. (2003). On embedding synfire chains in a balanced network. *Neural Computation*, 15(6):1321–1340.

Backofen, R., Borrmann, H. G., Deck, W., Dedner, A., Raedt, L. D., Desch, K., Diesmann, M., Geier, M., Greiner, A., Hess, W. R., Honerkamp, J., Jankowski, S., Krossing, I., Liehr, A. W., Karwath, A., Kloefkorn, R., Pesche, R., Potjans, T., Roettger, M. C., L. Schmiedt-Thieme, G. S., Voss, B., Wiebelt, B., Wienemann, P., and Winterer, V. H. (2006). A bottom-up approach to grid-computing at a university: the black-forest-grid initiative. *PIK - Praxis der Informationsverarbeitung*, 29:81–87.

Bastian, P., Birken, K., Johannsen, K., Lang, S., Neuss, N., Rentz-Reichert, H., and Wieners, C. (1997). Ug - a flexible software toolbox for solving partial differential equations. *Computing and Visualization in Science*, 1:27–40.

Binzegger, T., Douglas, R. J., and Martin, K. A. C. (2004). A quantitative map of the circuit of cat primary visual cortex. *J. Neurosci.*, 39:8441–8453.

Brette, R. (2006). Exact simulation of integrate-and-fire models with synaptic conductances. *Neural Computation*, 18(8):2004–2027.

Brette, R. and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.*, 94:3637–3642.

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., Diesmann, M., Morrison, A., Goodman, P. H., Harris, F. C., Jr., Zirpe, M., Natschläger, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A. P., Boustani, S. E., and Destexhe, A. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. *J. Comp. Neurosci.* Submitted.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comp. Neurosci.*, 8:183–208.

Churchland, P. S. and Sejnowski, T. J. (1992). *The Computational Brain*. The MIT Press, Cambridge, Massachusetts.

Denk, W. and Horstmann, H. (2004). Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biology*, 2(11):1900–1909.

Destexhe, A., Rudolph, M., and Pare, D. (2003). The high-conductance state of neocortical neurons in vivo. *Nat. Rev. Neurosci.*, 4:739–751.

Djurfeldt, M., Johansson, C., Örjan Ekeberg, Rehn, M., Lundqvist, M., and Lansner, A. (2005). Massively parallel simulation of brain-scale neuronal network models. Technical Report TRITA-NA-P0513, CSC, KTH, Stockholm.

Eberhard, J., Attinger, S., and Wittum, G. (2004). Coarse graining for upscaling of flow in heterogeneous porous media. *Multiscale Model. Simul.*, 2(2):269–301.

Einevoll, G. T., Pettersen, K. H., Devor, A., Ulbert, I., Halgren, E., and Dale, A. M. (2007). Laminar population analysis: Estimating firing rates and evoked synaptic activity from multielectrode recordings in rat barrel cortex. *J. Neurophysiol.* In press.

Fourcaud-Trocme, N., Hansel, D., van Vreeswijk, C., and Brunel, N. (2003). How spike generation mechanisms determine the neuronal response to fluctuating inputs. *J. Neurosci.*, 23:11628–11640.

Haeusler, S. and Maass, W. (2007). A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. *Cerebral Cortex*, 17:149–162.

Hammarlund, P. and Ekeberg, O. (1998). Large neural network simulations on multiple hardware platforms. *J Comput Neurosci*, 5(4):443–59.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans Neural Networks*, 14:1569–1572.

Jolivet, R., Lewis, T. J., and Gerstner, W. (2004). Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy. *J. Neurophysiol.*, 92:959–976.

Keat, J., Reinagel, P., Reid, R. C., and Meister, M. (2001). Predicting every spike: a model for the responses of visual neurons. *Neuron*, 30:803–817.

Kennedy, H. (2005). The DAISY Project. http://daisy.ini.unizh.ch.

Kumar, A., Schrader, S., Aertsen, A., and Rotter, S. (2007). The high-conductance state of cortical networks. *Neural Computation*. In press.

Latham, P. E., Richmond, B. J., Nelson, O. G., and Nirenberg, S. (2000). Intrinsic dynamics in neuronal networks. i. theory. *J. Neuophysiol.*, 83:808–827.

Lundqvist, M., Rehn, M., Djurfeldt, M., and Lansner, A. (2006). Attractor dynamics in a modular network model of neocortex. *Network: Computation in Neural Systems*, 17:253–276.

MacGregor, R. J. and Oliver, R. M. (1974). A model for repetitive firing in neurons. *Kybernetik*, 16:53–64.

Markram, H. and Peck, C. (2004). The Blue Brain Project. http://bluebrainproject.epfl.ch.

Marr, D. (1982). *Vision*. Freeman, New York.

Mehring, C., Hehl, U., Kubo, M., Diesmann, M., and Aertsen, A. (2003). Activity dynamics and propagation of synchronous spiking in locally connected random networks. *Biol. Cybern.*, 88(5):395–408.

Meier, K. (2005). The FACETS Project. http://facets.kip.uni-heidelberg.de.

Morrison, A., Aertsen, A., and Diesmann, M. (2007a). Spike-timing dependent plasticity in balanced random networks. *Neural Computation*. In press.

Morrison, A., Mehring, C., Geisel, T., Aertsen, A., and Diesmann, M. (2005). Advancing the boundaries of high-connectivity network simulation with distributed computing. *Neural Computation*, 17:1776–1801.

Morrison, A., Straube, S., Plesser, H. E., and Diesmann, M. (2007b). Exact subthreshold integration with continuous spike times in discrete-time neural network simulations. *Neural Computation*, 19(1):47–79.

Paninski, L., Pillow, J. W., and Simoncelli, E. P. (2004). Maximum likelihood estimation of a stochastic integrate-and-fire neural encoding model. *Neural Computation*, 16:2533–2561.

Potjans, T., Wiebelt, B., and Diesmann, M. (2007). Grid computing for computional neuroscience. In *10th Tamagawa-Riken Dynamic Brain Forum*, Hakuba, Japan.

Richardson, M. J., Brunel, N., and Hakim, V. (2003). From subthreshold to firing-rate resonance. *J. Neurophysiol.*, 89:2538–2554.

Rudolph, M. and Destexhe, A. (2006). Analytical integrate-and-fire neuron models with conductance-based dynamics for event-driven simulation strategies. *Neural computation*, 18(9):2146–2210.

Schutter, E. D. (1998). Dendritic voltage and calcium-gated channels amplify the variability of postsynaptic responses in a purkinje cell model. *Journal of Neurophysiology*, 80:504–519.

Senft, S. L. and Ascoli, G. A. (1999). Reconstruction of brain networks by algorithmic amplification of morphometry data. *Lecture Notes Computer Science*, 1606:25–33.

Sharp, A. A., O'Neil, M. B., Abbott, L. F., and Marder, E. (1993). Dynamic clamp: Computer-generated conductances in real neurons. *J. Neurophysiol.*, 69(3):992–995.

Tetzlaff, T., Morrison, A., Geisel, T., and Diesmann, M. (2004). Consequences of realistic network size on the stability of embedded synfire chains. *Neurocomputing*, 58–60:117–121.

Thomson, A. M., West, D. C., Wang, Y., and Bannister, A. P. (2002). Synaptic connections and small circuits involving excitatory and inhibitory neurons in layer 2–5 of adult rat and cat neocortex: Triple intracellular recordings and biocytin labelling in vitro. *Cerebral Cortex*, 12:939–953.

Tuckwell, H. C. (1988). *Linear Cable Theory and Dendritic Structure*, volume 1 of *Introduction to Theoretical Neurobiology*. Cambridge University Press, Cambridge.

Usui, S. (2003). Visiome: neuroinformatics research in vision project. *Neural Networks*, 16:1293–1300.

Usui, S., Yamaguchi, I., Ikeno, H., Takebe, K., and Okumura, Y. (2004). Visiome environment: enterprise solution for neuroinformatics in vision science. *Neurocomputing*, 58–60:1097–1101.

van Schaik, A., Cohen, A., Indiveri, G., Hasler, P., Etienne-Cummings, R., Douglas, R., Shamma, S., Sejnowski, T., and Horiuchi, T. (2006). The telluride workshop. http://ine-web.org/telluride-conference-2006/telluride-overview/index.html.

Vogels, T. P. and Abbott, L. F. (2005). Signal propagation and logic gating in networks of integrate-and-fire neurons. *J. Neurosci.*, 25:10786–10795.

Wittum, G. (2007). μG. http://sit.iwr.uni-heidelberg.de/~ug/.

www.incf.org

INCF Secretariat
Karolinska Institutet
Nobels väg 15 A
SE-171 77 Stockholm
Sweden

Tel: +46 8 524 87 093
Fax: +46 8 524 87 150
E-mail: info@incf.se

incf International Neuroinformatics Coordinating Facility