BLISS: Towards the simulation of brain-like systems

Marc-Oliver Gewaltig, Andreas Richter, Rüdiger Kupper

2002

Preprint:

This is an accepted article published in Neurocomputing. The final authenticated version is available online at: https://doi.org/[DOI not available]

In *Neurocomputing, 44-46:805-810, 2002* .

BLISS: Towards the Simulation of Brain-Like Systems

Marc-Oliver Gewaltig [a], Andreas Richter [a], Rüdiger Kupper [b]

[a] *Future Technology Research, Honda R&D Europe (Deutschland) GmbH, D–63073 Offenbach, Germany*

[b] *AG Neurophysik, Philipps-University, D–35032 Marburg, Germany*

Abstract

BLISS is a novel framework for simulating large, structured neuronal systems. It is designed to investigate the functional behavior of neuronal systems in the context of anatomical, morphological, and electrophysiological properties. BLISS aims at large networks (10^5 neurons), while maintaining biological detail. This is achieved by combining a broad range of abstraction levels in a single network simulation. Great biological detail is then only maintained at the points of interest, while the rest of the system is modeled by more abstract components. Here, we describe the conception of BLISS and show example simulations to illustrate its key features.

Key words: large-scale modeling, structured networks, event driven simulation

1 Introduction

In computational neuroscience, simulations are used to investigate models of the nervous system at functional or process levels. Consequently, a lot of effort has been put into developing appropriate simulation tools and techniques and a plethora of simulation software, specialized on single neuron or small sized networks (e.g. [1,3]) is available.
Recently, however, there is growing interest in large scale simulations, involving some 10^4 neurons while maintaining an acceptable degree of biological detail. Thus, there is need for, possibly parallel, simulation software which supports such simulations in a flexible way (see also [2]).
In this contribution we describe a simulation system which is designed to simulate large-scale models of the nervous system. The system is open and

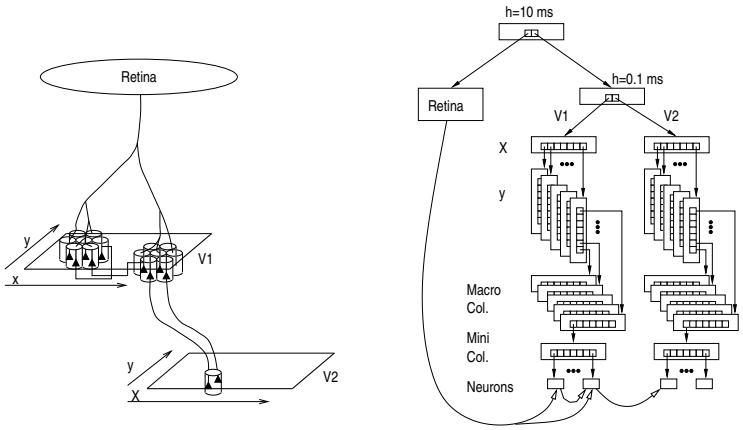

Fig. 1. Example of a structured network model (left) and how it is represented (right).

extensible and provides support for structured networks of heterogeneous elements at different description levels.

2 Modeling strategy

A BLISS simulation tries to maintain a close correspondence to an electrophysiological experiment. There are three conceptual domains which have to be mapped to the software level: (1) the description of the neuronal system, that is the network components and the network, (2) the description of the experimental setup, and (3) the experiment protocol.

The standard approach to neural modeling is to consider neurons as the basic network components and describe the network in terms of the neurons, their positions and their projections. This approach can be called bottom up, since the network is built from its basic constituents upwards.
However, the brain is a heterogeneously structured system, composed of numerous areas and nuclei which are distinguished by their anatomical, physiological, and functional properties. Moreover, each of these parts has a specific architecture which is to a large extent genetically determined. The structure of the brain, thus, conveys information which is beyond the mere connectivity (i.e. synaptic connections) of its constituents.
Thus, we adopt a top-down approach (see also [4]): Networks are regarded as hierarchical structures which may be represented by trees. A network is described in terms of abstract components which may either be *atomic* or *compound*. Atomic network elements may range from sub-neuronal compartments to large populations of neurons, and may, thus, differ considerably in complexity and degree of abstraction. Compound network elements are defined in terms of other elements and their mutual connections. They may be nested and may, thus, be used to define concepts like areas, macro-columns,

and minicolumns. Structural model concepts can therefore be mapped to the simulated network.
Fig. 1 (left) shows an example of such a hierarchically structured model (e.g. [5]). The model network consists of several sub-structures, described at different levels of abstraction: a *retina* and two model brain areas, *V1* and *V2*. The retina sends input to a first visual area V1 which consists of topographically organized macro-columns. These, in turn, consist of orientation columns. Finally, orientation columns consist of model neurons. The other area, V2, has a similar structure. The right Panel shows how the model may be represented in the framework presented here. The tree reflects the *semantic structure* of the model. Synaptic connections are omited for simplicity. Atomic elements are represented by separate objects as leafs of the network tree. Examples of atomic elements are the neurons in a column or the model retina which models the behaviour of a large number of identical neurons in a single element. The model areas V1 and V2 and the various columns are examples for compound elements.

In an electrophysiological experiment, a number of different devices are used to stimulate and observe the neuronal system. In our approach, the measurement and stimulation process is an explicit part of the model. Accordingly, devices are explicitly mapped to the simultion in order to arrive at *virtual experiments*.

3 Structure of BLISS

At the top level, BLISS consists of a number of independent modules. BLISS is designed in an object-oriented style (C++). An abstract base class defines a core interface which has to be implemented by each module. The various modules are combined by a module loader, which is part of a simulation language interpreter (SLI). The interpreter is not only the primary interface to the user, but it also provides a set of basic data-structures for data exchange between modules.
The simulation language interpreter is a simple stack machine, optimized for the fast processing of high-level commands and objects.
The second important component of BLISS is the simulation kernel. It implements all functionalities to define and simulate large structured networks of biologically realistic neural networks. It provides

(1) base classes for neuronal models,
(2) derived implementations of important models,
(3) a network driver class to manage the temporal update of all simulation elements,
(4) a SLI module to provide high-level access to the kernel, and

(5) a SLI module to provide a high-level interface to the available models.

Items 1 through 3 are combined in a single C++ library which may also be used without the simulation language interpreter.
In addition to the simulation kernel, there are a number of other modules which provide important functionality. Some modules solve well define mathematical problems like differential equation, some perform high-level operations on arrays and lists or provide random number generators. Other modules implement interfaces to the host operating system or on-line help facilities.

4 The simulation kernel

BLISS uses an object-oriented approach. The system to be simulated is conceptually broken down into *elements* like neurons, synapses, columnar circuits, or whole cortex areas, which are mapped to *classes* at the C++ level. A simulation is build from the set of pre-defined elements, by dynamically (during runtime) combining them to represent the desired neuronal system.
Elements share a common interface, but may differ considerably in their purpose, functionality, or level of abstraction. Different element-types, ranging from sub-neuronal components (e.g. compartments) to supra-neuronal elements (e.g. populations of neurons), may co-exist in the same network.
While elements are *hard coded* in C++, their configuration is usually not. Rather, configuration is done at the level of the simulation language. Simulations are defined by short scripts and do not require re-compilation of the main executable. The simulation, however, runs without the overhead of an interpreter.

4.1 Model hierarchy

Computational neuroscience is faced with the problem that there is no standard model neuron; nor is their agreement at which scale neuronal systems should be investigated. This is reflected in the class hierarchy of network elements which are currently used in BLISS.
The range of possible abstractions is covered by deriving several abstract base classes from the top-level base class, each providing a starting point for a hierarchy of models at a particular description level (Fig. 2).
The element base class defines the minimal interface that each element has to implement.
Since neuron models have different configuration requirements, BLISS provides a named parameter interface, allowing the use of meaningful names like *Resistance* or *Capacity* for the parameters.

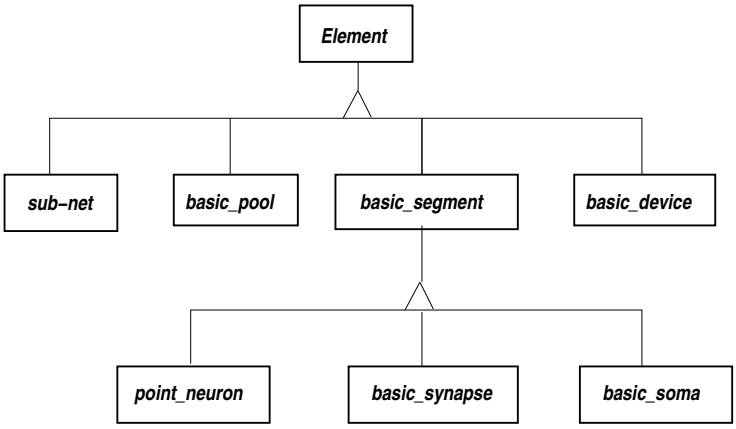

Fig. 2. Simplified class hierarchy for network elements.

4.2 Networks and elements

Network elements fall into two categories: atomic elements and compound elements. The most important compound element is called sub-network. It is used to group arbitrary elements and may be nested. Functionally related elements can be grouped into circuits which may themselves be grouped into higher order circuits. Thus, a network can be thought of as tree with a root sub-network at its base. To the user the network tree is comparable to a directory tree in a file system: Sub-networks correspond to *directories* and atomic elements to *files*. The driver provides functions to operate on the network tree similar to the UNIX file-system commands.

4.3 Connections and communication

Connections allow communication between elements. They may be stored explicitely as target lists, or algorithmically. Since in large networks, the connectivity information dominates the memory consumption, algorithmic connections are an efficient representation of stereotypic connectivity patterns (e.g. *one to all* or *nearest neighbor* couplings).

During the simulation, elements exchange information in form of discrete events. While simple neurons may just exchange point events (spikes), more powerful element-types need to exchange more complex information. Thus, BLISS a generic event mechanism which supports the efficient exchange of arbitrarily complex objects.

However, BLISS does not use an *event-driven* paradigm. A large class of neuron-models communicate by exchanging spikes as point-events. However, each generated event is sent to all neuronal targets, resulting in an forbiddingly large number of events per spike (one for each target). Here, a time-

locked update allows for an optimized delivery, effectively avoiding this event replication. Moreover, event-driven update schemes, require knowledge about the distribution of transmission delays, which might not be easily accessible in the case of algorithmic connections.

5 Discussion

The concepts described above have been implemented (see e.g. [6], this volume, for an application). Clearly, it can not be expected that all design goals are reached at once and thus, further work needs to be done to improve memory consumption and processing speed.
The development of simulation methods is a taxing exercise for small research groups, because deviates resources from the main research activities. Here, a collaboration offers a number of advantages: first, the entire task of designing, implementing, testing, and documenting simulation software is distributed over a wider community, reducing the work-load for each party significantly. Second, each group benefits from synergetic effects which come with the joint efforts of a larger team. Thus the system described here is now developed in collaboration with the research groups of A. Aertsen (Freiburg), T. Geisel/M. Diesmann (Göttingen), R. Eckhorn/R. Kupper (Marburg), H-E. Plesser (Ås, Norway), and Honda R&D Europe.

References

[1] James M. Bower and David Beeman. *The Book of GENESIS: Exploring realistic neural models with the GEneral NEural SImulation System*. TELOS, Springer-Verlag, New York, 2nd edition, 1997.

[2] N. Goddard, G. Hood, F. Howell, M. Hines, and E. De Schutter. NEOSIM: Portable large-scale plug-and-play modelling. In James. M. Bower, editor, *Computational Neuroscience: Trends in Research 2001*, pages 1657–1661. Elsevier Science, Amsterdam, 2001.

[3] Michael Hines and N. T. Carnevale. The NEURON simulation environment. *Neural Computation*, 9:1179–1209, 1997.

[4] P. Johnston, F. Hensken, and W. McGowan. NEU-MODEL: A multi-level object-oriented dynamic emulation laboratory. In James. M. Bower, editor, *Computational Neuroscience: Trends in Research 2001*, pages 1671–1677. Elsevier Science, Amsterdam, 2001.

[5] Edgar Körner, Marc-Oliver Gewaltig, Ursula Körner, Andreas Richter, and Tobias Rodemann. A model of computation in neocortical architecture. *Neural Networks*, 12(7–8):989–1005, 1999.

[6] H. E. Plesser, G. T. Einevoll, and P. Heggelund. Mechanistic modeling of the retinogeniculate circuit in cat. In James. M. Bower, editor, *Computational Neuroscience: Trends in Research 2002*. Elsevier Science, Amsterdam, 2002.