Chapter 6

Working with NeuroSim

The time required to prepare a simulation based on experimental results is often a critical factor in scientific research. Computational modelling is therefore becoming essential in neuroscience research. Thus, one of the characteristics of computer simulation programs that is of great importance to neurobiologists is a clear and intuitive user interface.

This chapter gives a description of all aspects of working with NeuroSim. The most important aspects are the definition of the model, the assignment of model element properties, and the visual presentation of simulation results. The user does not need to deal directly with code describing mathematical calculations, but can instead prepare the simulation via a friendly graphical user interface. The GUI for the modeler is provided with the client and does not require any additional resources from the standard computer.

The client is written in Java and can be started either as an application on the user's computer regardless of the operating system, or as an Java applet from the Internet.

6.1 Getting started

After starting NeuroSim, the main application window appears on the screen, and the user can set up the simulation process and control the simulation results (see Fig. 6.1). The window's main elements are: the main menu, the main panel for presentation of neuron's structure, the toolbar kit, and the status bar.

The main menu contains the commands for model definition and control of the simulation process. The most often-used actions can also be performed using the toolbar kit, on which the command buttons for quick access are located.

The status bar shows the status of the simulation process: time and simulation parameters.

The GUI was designed using the Swing components of the Java Toolkit. It enables incorporation of high-quality 2D graphics for the resulting data representation, and provides the ability for drag-and-drop operations for model

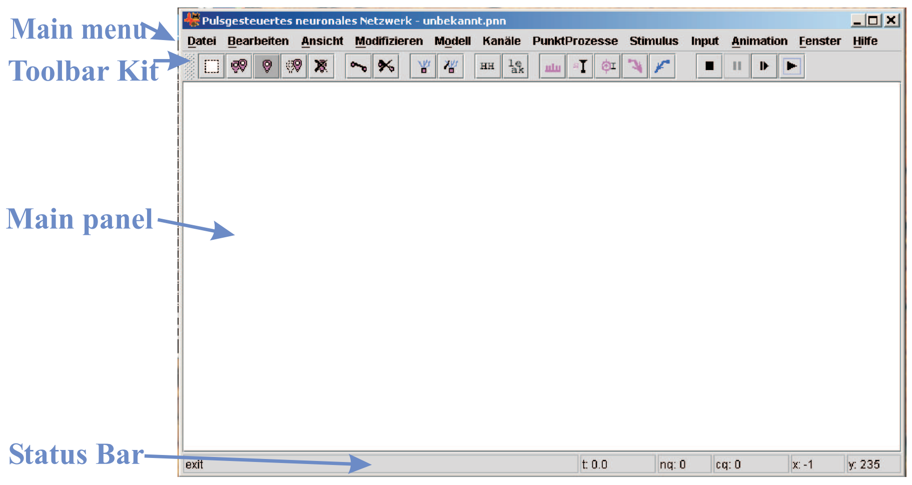


Figure 6.1: NeuroSim main window.

definition and structure changing.

The application is internationalized, so that text elements, such as status messages and labels of the GUI components, can be dynamically loaded in different languages. Thus, the application can be adapted to various languages and regional settings without engineering changes. In Fig. 6.1, the GUI elements of the NeuroSim main panel are adjusted for German users, while Fig. 6.5 represents labels and menu captions in English. The characteristics of the program are stored in the “translator files” and used after identification of a particular language and country.

The parameters of the GUI elements from the last session, such as the size and position of the displayed windows, are saved in a file in the user’s home directory and are kept for the next sessions.

The Applet version of NeuroSim is capable of using the local system resources, where the description of GUI elements and model definition files are stored. For this purpose, the applet was signed to allow access to local system resources defined by the user and based on the system’s security policy.

6.2 Model definition

The type of model and mode of operation can be chosen in the main menu. New elements for the model can be added to the simulation just by choosing the elements in the toolbar kit (or in the main menu) and then defining their positions on the main panel by using the mouse. The type of model has to be set before one starts to define the elements of the model. That is done via the “Model” menu, which is shown in Fig. 6.2.

The available model types can be divided into two groups: pulsed neural

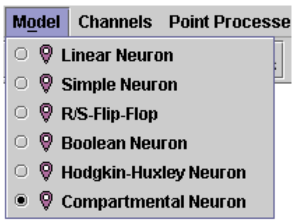


Figure 6.2: NeuroSim "Model" menu.

networks (see Section 2.2.3) and real conductance-based neural networks (see Section **??**). The following models belong to pulsed neural networks: *Linear Neuron* model, *Simple Neuron* model (or *Spike Response Model*), *RS-Flip-Flop* model, and *Boolean Neuron* model. The linear neuron model consists of neurons whose activities are changed linearly. The Spike Response Model has been described in the review of neural models contained in Chapter 2. The RS-Flip-Flop model contains neurons whose activity changes in the same manner as the RS-Flip-Flop elements. Two outputs of the start neuron, "Q" and "NQ", and two inputs, "R" and "S", of the end neuron are used as parameters of the connection between RS-Flip-Flop neurons. Boolean neuron activity depends on the combination of inputs from connected neurons, that is, inputs described with "logical" properties of the neuron, which can be "OR", "AND", "NAND", "NOR", or "XOR".

Due to the relative simplicity of pulsed neural networks, their simulation is performed on the local computer, whereas the simulation of real biological networks is realized as an application based on the client-server architecture. The server is responsible for intensive calculations arising as a mathematical result of detailed simulations. In real, conductance-based neural networks, it is possible to simulate neurons that contain ionic channels with Hodgkin-Huxley dynamics ("Hodgkin-Huxley Neuron") as a default, or to define the compartmental structure of each neuron and all elements the neurons are composed of ("Compartmental Neuron"). Elements such as voltage-gated channels, synaptic channels, point processes, and stimuli can be added to the model by using either the main menu or the toolbar.

The buttons in the "Modify" menu and the corresponding buttons in the toolbar are used for changing the mode of the next operation, as shown in Fig. 6.3. The mode of operation can be: adding a new neuron, deleting a neuron, changing the position of a neuron, marking neurons for subsequent copy, copy of marked elements, establishing a new connection, deleting a connection, adding a new dendritic compartment, or deleting a dendritic compartment.

After choosing the mode of operation, one can define or change the model with the drag-and-drop operations. When the mode "add neuron" is chosen, new neurons are added onto the main panel by clicking the left mouse button. The neurons can be moved by clicking the left mouse button and dragging (see

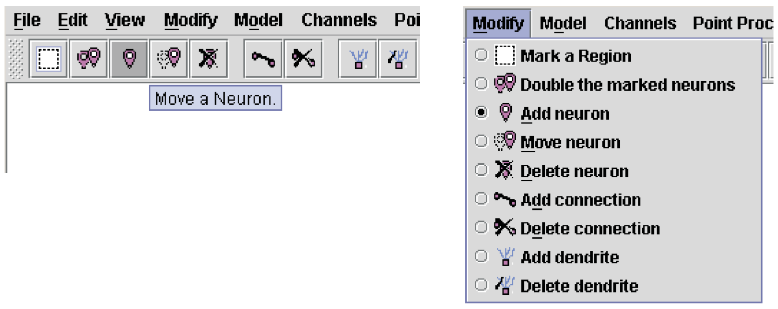


Figure 6.3: "Modify" menu and toolbar buttons for changing the editing mode.

Fig. 6.4(a)). When the neuron is moved, its contour and connection contours are represented with thin lines. When the mode "add connection" is chosen, the connections can be established by choosing the starting neuron and dragging with the left mouse button held until the last neuron is reached (see Fig. 6.4(b)).

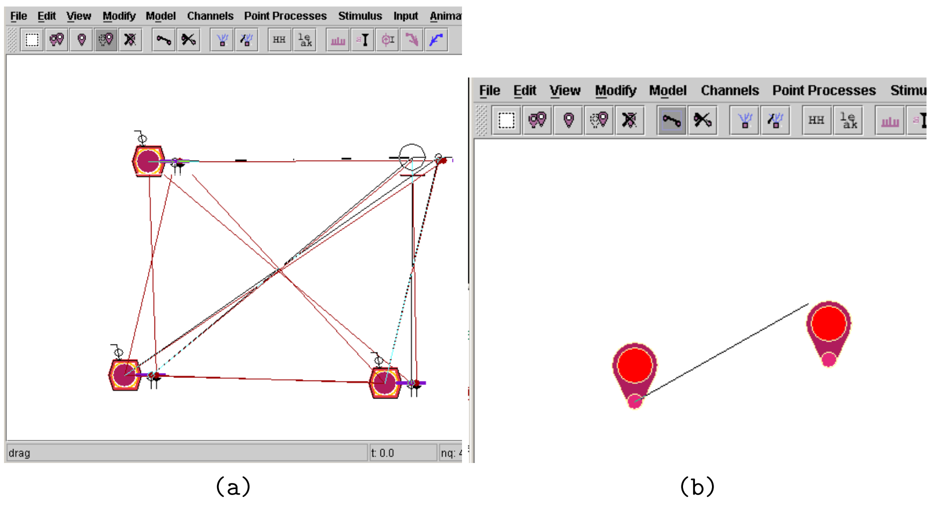
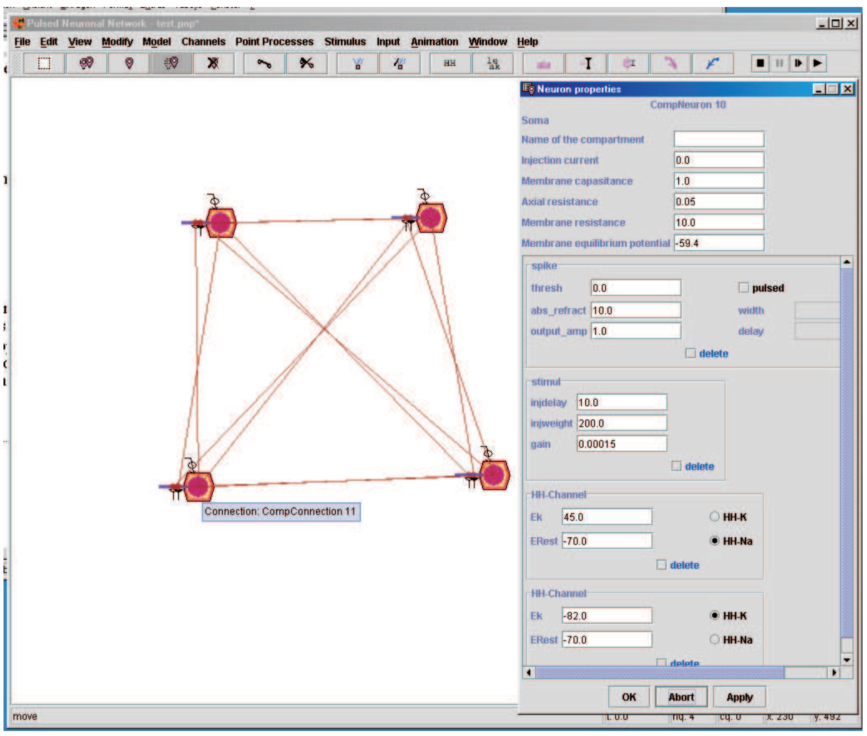


Figure 6.4: Editing of the model. a) Changing the position of the compartmental neuron; b) Establishing a connection between spiking neurons (Spike Response Model).

Neurons with the same physiological and anatomical parameters can be copied, which simplifies the process of model definition.

For the compartmental neural model, the parameters of all elements (neurons, compartments, ionic and synaptic channels, neuron connections, etc.) can be viewed in separate windows (see Fig. 6.5). The properties window of the neurons or connections can be made visible by a right mouse click on the element. The properties window provides the possibility to edit the properties of neural elements.


Pulsed Neuronal Network - test.pnp*
File Edit View Modify Model Channels Point Processes Stimulus Input Animation Window Help
Connection: CompConnection 11
move
OK
Abort
Apply

Neuron properties

CompNeuron 10

Soma

Name of the compartment	
Injection current	0.0
Membrane capasitance	1.0
Axial resistance	0.05
Membrane resistance	10.0
Membrane equilibrium potential	-59.4

spike

thresh	0.0	☐ pulsed	
abs_refract	10.0	width	
output_amp	1.0	delay	

☐ delete

stimul

injdelay	10.0
injweight	200.0
gain	0.00015

☐ delete

HH-Channel

Ek	45.0	HH-K
ERest	-70.0	[✓ HH-Na]

☐ delete

HH-Channel

Ek	-82.0	[✓ HH-K]
ERest	-70.0	HH-Na

☐ delete

Figure 6.5: Compartmental neuron structure in the main window and parameters of model elements in a separate window.

Buttons of the menus “Channels”, “PointProcesses”, “Stimulus”, and “Input” are used for including ionic channels that can be voltage-gated or synaptic, point processes, injected currents, and stimuli from random or pulsed-spike generators into the model.

Thus, all necessary elements can be defined in the model via a graphical user interface, and no extra programming code needs to be written for model definitions.

6.2.1 Example: network of neurons with Hodgkin-Huxley channels

Now let us look at some examples of how the model can be defined with the NeuroSim GUI. I will start with the neural network model containing voltage-gated Hodgkin-Huxley channels, since the Hodgkin-Huxley model still remains the formalism that is most often chosen by neuroscientists.

I will illustrate the simulation of a network consisting of two realistic neurons in a feedback configuration. Each neuron is composed of two compartments corresponding to a soma and a dendrite. The soma in each compartment is composed of two variable conductance ionic channels for sodium (Na^+) and potassium (K^+), which are described with Hodgkin-Huxley dynamics. The dendritic channels are synaptically activated and responsible for the cell connections. Spike events of each cell reach their destinations after a specific delay. After the spike arrival, the synaptic channels are activated. The conductance change upon activation of the synaptic channels is described with an *alpha function*

$$G_{syn}(t) = \frac{const}{\tau_1 - \tau_2} \cdot (\exp(-t/\tau_1) - \exp(-t/\tau_2)) \tag{6.1}$$

The type of model for this simulation should be defined as “Compartmental Neuron”. By choosing the operation mode “Add a neuron”, a neuron with a soma and one dendritic compartment as a default will be added . For neurons with a compartmental structure, it is possible to edit the parameters of a single compartment by clicking on compartment’s visual representation.

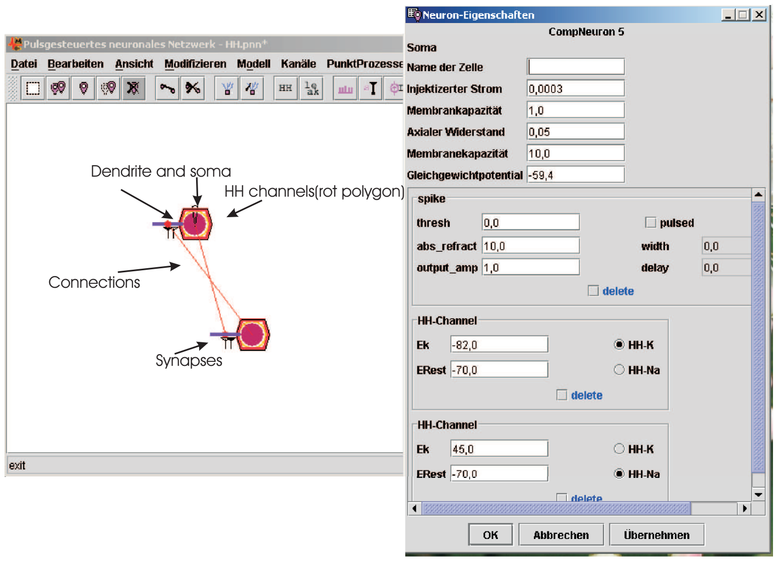
The next step in model definition is adding the Hodgkin-Huxley channels to a compartment corresponding to a soma, which can be done by activating the button “Hodgkin-Huxley channel” from the “Channels” menu and clicking on the compartment “soma”. The new polygonal symbol will be drawn around the soma and the properties of the added channel will be represented in the properties window for the soma. In our model, the soma consists of two Hodgkin-Huxley channels for sodium and potassium. The type of the channel and its properties can be changed in the properties window (see Fig. 6.6).

Injected currents can be defined either as a constant injection current or as a pulsed current. A constant injection current is set by changing the `injection` property of a compartment, which can only possess positive values. A pulsed current can be added after activation of the “Pulsed current” button in the menu “Stimulus”. In our model, a constant current is injected into the soma of the first neuron.

To establish a connection between two biological cells, one needs to add the element “spike generator” into the presynaptic cell and the element synaptic channel into the postsynaptic cell. The modes for these operations are available from the toolbar or from the menu. Then, after choosing the mode “Add a connection”, a mouse drag will draw a new connection. The presence of the spike generator and the synaptic channel will be controlled upon setting the new connection.

In Figure 6.6 the structure of our model is presented. Here, one can observe the neural network structure, as well as details of the model elements, e.g. the compartments, the voltage- and synaptically-activated channels. A separate window for editing the properties of the compartment “soma” is also shown as an example.

The injection current into the soma of the first cell activates the Hodgkin-Huxley channels and corresponds to spike initiation in the first cell. The spike event will activate the excitative synaptic channel in the dendrite of the second


Pulsgesteuertes neuronales Netzwerk - HH.pnn*
Datei Bearbeiten Ansicht Modifizieren Modell Kanäle PunktProzesse
Dendrite and soma
HH channels(rot polygon)
Connections
Synapses
exit
Neuron-Eigenschaften
OK
Abbrechen
Übernehmen

CompNeuron 5

Soma

Name der Zelle	
Injektizerter Strom	0,0003
Membrankapazität	1,0
Axialer Widerstand	0,05
Membranekapazität	10,0
Gleichgewichtpotential	-59,4

spike

thresh	0,0	☐ pulsed	
abs_refract	10,0	width	0,0
output_amp	1,0	delay	0,0

☐ delete

HH-Channel

Ek	-82,0	● HH-K
ERest	-70,0	○ HH-Na

☐ delete

HH-Channel

Ek	45,0	○ HH-K
ERest	-70,0	● HH-Na

☐ delete

Figure 6.6: The structure of a two-cell model with Hodgkin-Huxley channels.

cell. It gives the stimulus for activation of the Hodgkin-Huxley channels in the soma of the second cell, which accordingly activates the inhibitory synaptic channel in the dendrite of the second cell.

After performing the simulation on the server, it is possible to run an animation of the results. The spike events in the main window displaying the structure of the model can be observed, and the graphs of voltage spread and synaptic channel conductance variation are represented in the graphical windows (see Fig. 6.7).

6.2.2 Example: passive dendritic cable

For the next example, let us consider how the model of the realistic neuron containing several dendritic compartments can be simulated with NeuroSim. The results of the simulation show a change in membrane voltage in the passive dendritic tree as has been described in Section 3.2.2.

The dendrites are added to a neuron modelled with the compartmental approach after setting the “Add a dendrite” mode from the menu or toolbar. A new dendritic compartment can then be drawn by left-clicking on an existing compartment. A compartment can be deleted by choosing the “Delete a dendrite” mode and clicking on the compartment that is located on the end of the dendritic cable.

The neuron that consists of a soma compartment and 10 identical dendritic compartments was constructed in the manner described above. The soma compartment is modelled as a spherical compartment with a diameter

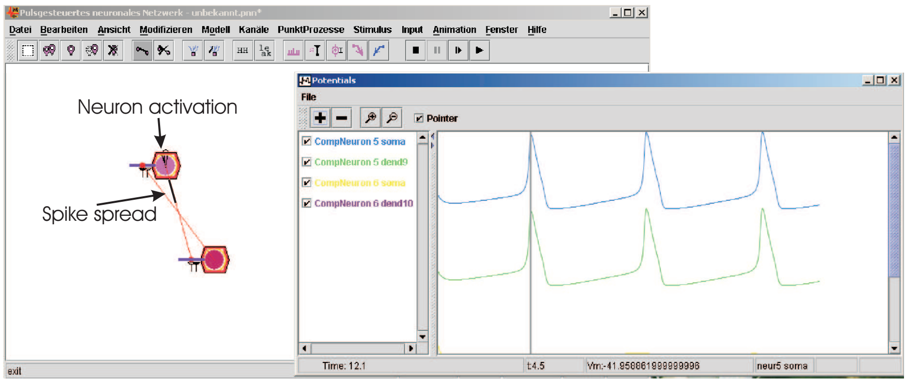


Figure 6.7: The animation of simulation results of a two-cell model with Hodgkin-Huxley channels.

$d = 0.005\ cm$, while the dendritic compartments have a cylindrical form with length $l = 0.0001\ cm$ and diameter $d = 0.005\ cm$. Each compartment has electrical properties: specific membrane resistance $R_M = 10\ k\Omega \cdot cm^2$, specific membrane capacitance $C_M = 1\ mF/cm^2$, and specific axial resistance $R_A = 0.1\ k\Omega \cdot cm$. The properties are adjusted in the properties windows for each compartment.

One can, for example, define a constant injection current $I = 0.0002\ \mu A$ for the soma. This can be made using the properties window of the soma compartment. Figure 6.8 shows the screenshot of NeuroSim with the main window of the constructed model and the results of the simulation.

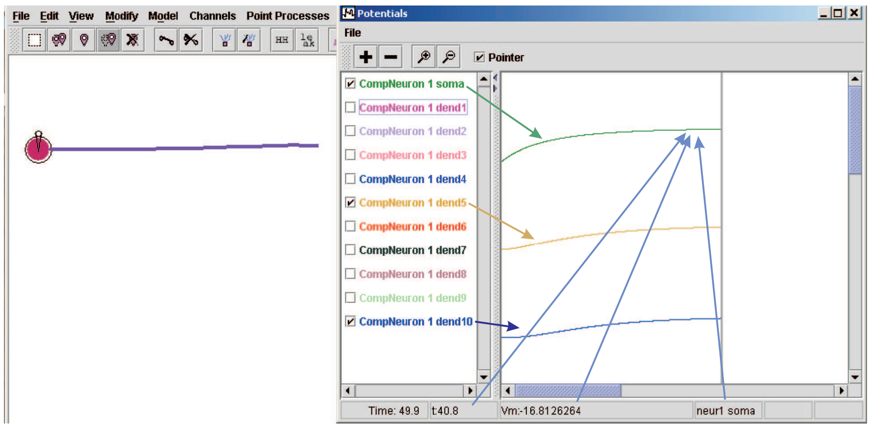


Figure 6.8: Passive cable model with constant current injection.

Using the compartment's model parameters, one can calculate the parameters, which influence on the form of the voltage change solution (see Section 3.2.2). One can see that each compartment has a length of 0.1 λ (λ is a space constant) and electronic length $L = l/\lambda$ of the cable, with $L = 1$.

By analyzing the model structure, one can see that no current can flow to the right from the last dendritic compartment. Therefore, the model corresponds to the "sealed end" boundary conditions; the solution of the equation for voltage change at a certain cable position is shown in Fig. 3.9.

One can also consider the case of a pulsed injection with a current pulse width of 30 ms and time between pulses of 10 ms. The results of a simulation for this case are shown in Fig. 6.9. The current during the pulse will influence the voltage of the input compartment by increasing its magnitude; after that one can observe its decline.

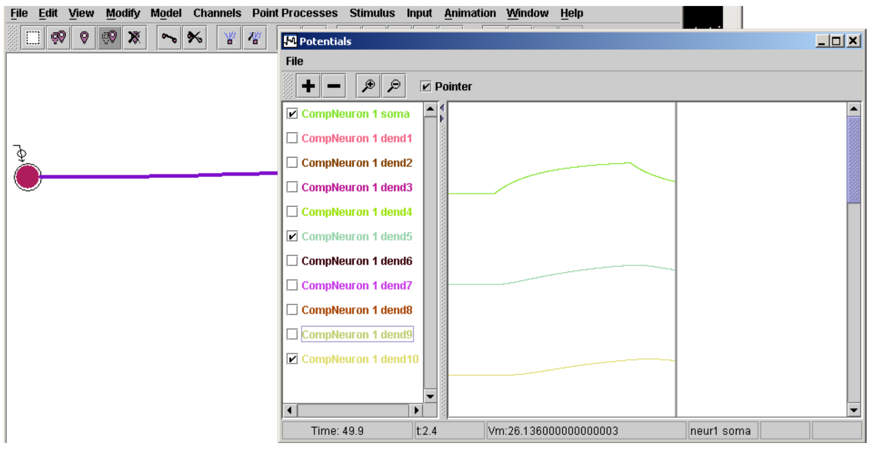


Figure 6.9: Passive cable model with pulsed current injection.

The next experiment shows how pulsed synaptic input can be integrated into the soma compartment. This operation is made using the pulsed spike generator and synaptic channel elements that provide pulsed synaptic input. For this, one needs to activate the corresponding buttons from the menu and clicking on the soma. In the example, I have used synaptic channel of the excitatory type, corresponding to sodium. The pulsed input has a delay of 10 ms and a width of 10 ms. The synaptic channel is activated with the connection weight, changing the "gain" property. Simulation results in this case in the curves are shown in Fig. 6.10.

One can continue modelling by constructing a neuron with a more complex dendritic structure, as for example shown in Fig. 6.11.

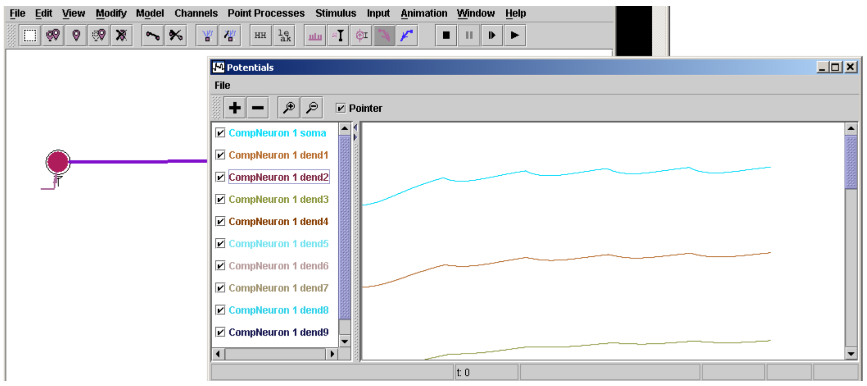
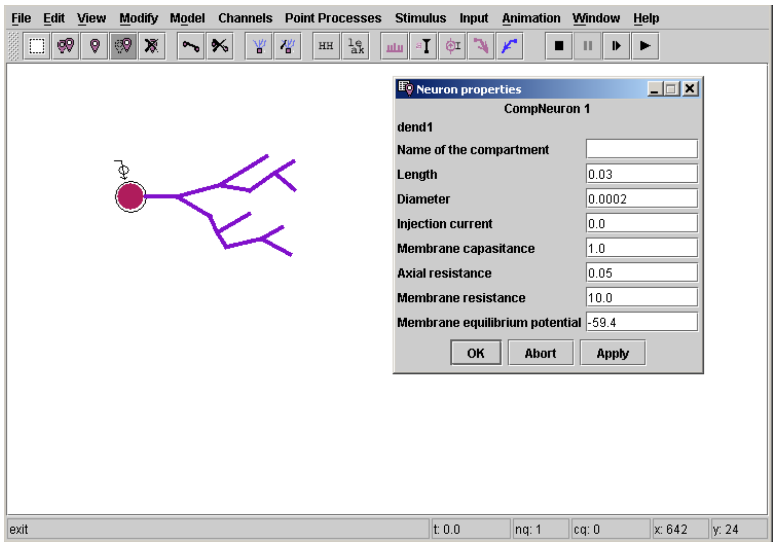


Figure 6.10: Passive cable model with pulsed synaptic input.


File Edit View Modify Model Channels Point Processes Stimulus Input Animation Window Help
Neuron properties
OK
Abort
Apply
exit
t: 0.0
nq: 1
cq: 0
x: 642
y: 24

CompNeuron 1

dend1

Name of the compartment	
Length	0.03
Diameter	0.0002
Injection current	0.0
Membrane capasitance	1.0
Axial resistance	0.05
Membrane resistance	10.0
Membrane equilibrium potential	-59.4

Figure 6.11: Compartmental neuron with several dendrites constructed with the NeuroSim GUI.

6.3 Model description files

The prepared models can be saved onto a hard disk in two formats: the standard XML model description language (defined as *.pnp format for our program) or as files containing the state of the model in the form of the serializable objects, since all neural elements are provided with the serialization facility (*.pnn). Operations of saving and opening model description files are performed from the "File" menu, as shown in Fig. 6.12.

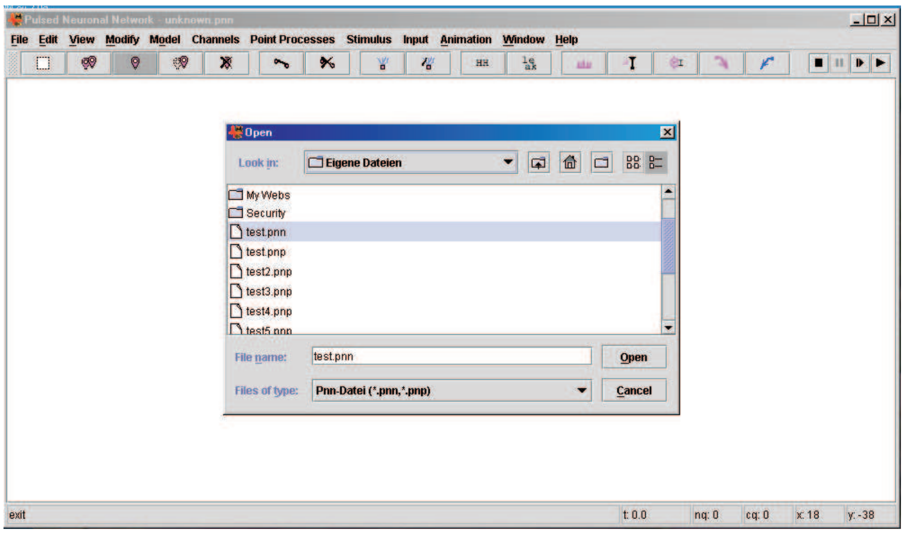


Figure 6.12: Opening and saving model descriptions in files.

One can describe neural models in the XML language format, a text-based markup language that has become a standard for data exchange. The advantage is that XML documents are easily processed and have a hierarchical structure. Detailed neural models containing a number of neurons in a compartmental structure and their connections can be logically presented in XML format as a file with a hierarchical structure. It is also important that a number of transformation tools are available.

The transformation of documents into XML format has been realized using the Java API for parsing XML documents – SAX 2.0 (Simple API for XML). SAX processing consists of two steps: creating a content handler and invoking the parser with a designated content handler. The registered parser will invoke redefined methods of a content handler, which receives some of the logical elements of the parsing document, as, for example, a new start tag, an end tag, or actual character data. Thus, using these methods, I have defined the program's response to all of the tags that can be found in the neural description documents. Neural elements will be initialized and added to a new simulation model with the properties selected from the parsing content. The parsing rules are described with methods of a content handler.

If one saves the model properties in XML format, a description is generated for each of the model elements. Since XML data is a plain text, a string describing each object is generated and will be added to the XML file.

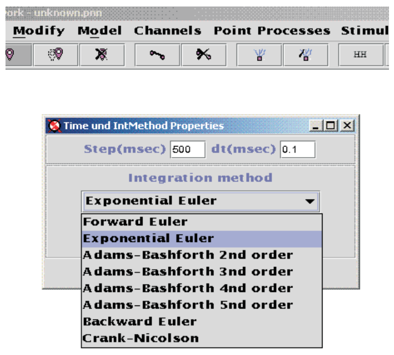
The models can be saved directly at their definition if the simulation on the server has not yet been performed and the client does not contain any data for animation. In this case, the XML files consist of a description of the compartmental structure of neurons, conductive channels, point processes, activated stimuli, and neural connections. When the results of the simulation from the server are also available, the results of the simulation will be saved in separate files. An XML file will then include a tag with the names of the files containing simulation results.

The second file format is possible since all objects that can make up the simulation models have been implemented as serializable objects provided by the Java serialization facility. This property provides the possibility to create an archival copy of an object for use in another instance of the same program. Serialized objects are saved as a byte data stream, which contains a summary of all information about its fields and methods instead of separate information about each field. They can be read and appended using the class of byte streams designed for this operation.

6.4 Connection to the server

The simulation of the defined model is performed on the remote server. For that, the client program automatically creates the XML description of the model, which defines the set of differential equations to be solved on the server. These data is sent to the server.

The connection to the server can be established from the "Animation" menu after setting up the method and size of numerical integration step (see Fig. 6.13).


ork - unknown.pnn
Modify
Model
Channels
Point Processes
Stimul
Time und IntMethod Properties
Forward Euler
Exponential Euler
Adams-Bashforth 2nd order
Adams-Bashforth 3nd order
Adams-Bashforth 4nd order
Adams-Bashforth 5nd order
Backward Euler
Crank-Nicolson

Step(msec) 500 dt(msec) 0.1

Integration method

Exponential Euler

Figure 6.13: Setting up the method and size of numerical integration step.

The connection to a server running Genesis or to our own C++ server can be established via the menu buttons "Start with Genesis" or "Start with C-server". The Genesis simulator provides the possibility to use numerical methods ranging from the crude explicit forward Euler method to highly stable implicit methods [49], [29]. Numerical integration on the C++ server is performed with the Euler exponential method. The user can interactively define the step size and interval of the simulation to adjust the simulation's speed and accuracy.

6.5 Simulation results

The results of a simulation are uploaded to the user's computer automatically. The graphic output of membrane voltage and channel conductance can be represented visually. The graphs can be observed in the windows with scrolling panels and a list of optional visualization variables (see Fig. 6.14).

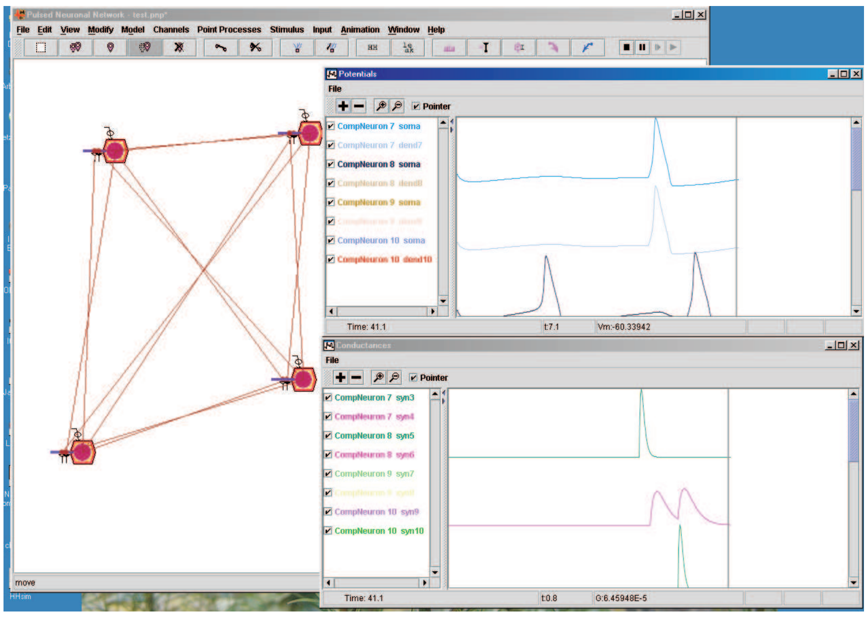


Figure 6.14: Visualization of simulation results.

6.5.1 Animation of the resulting data

The network simulation can be played back as an animation showing the neuron activations and the pulses propagating through the network connections (see Fig. 6.15). When the neuron is active, it is marked with a pink color; spike propagations are represented as moving strokes.

The animation was implemented as a thread with two modes: asynchronous and synchronous. The asynchronous mode allows execution of a task from an action event handler in a background thread, so that the GUI remains responsible. Synchronous mode allows one to perform the periodic actions with just one thread. These modes were developed with *SwingWorker* and the *Timer* Java utility classes, correspondingly [13].

The "Time properties" window activated from the "Window" menu (Fig. 6.16) allows simulation mode changes as well as the interval between updates and number of updates setting.

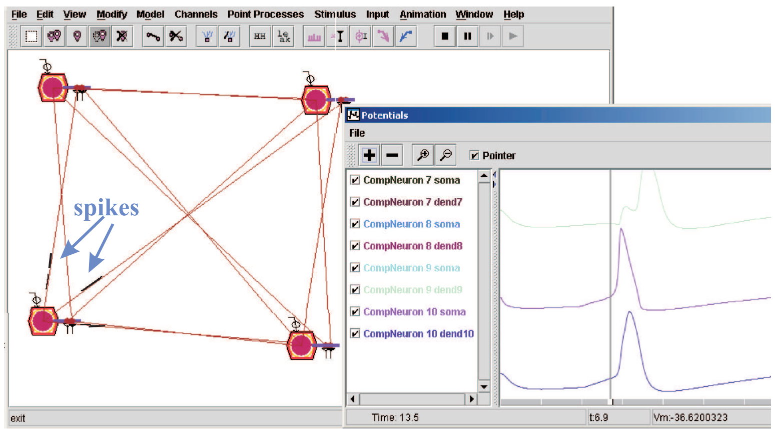
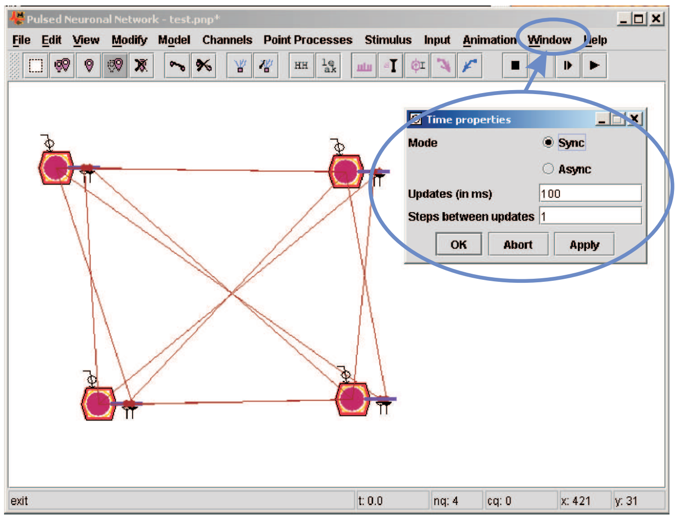


Figure 6.15: Animation of the network simulation in NeuroSim.


Pulsed Neuronal Network - test.pnp*
File Edit View Modify Model Channels Point Processes Stimulus Input Animation Window Help
Time properties
OK
Abort
Apply
exit
t: 0.0
nq: 4
cq: 0
x: 421
y: 31

Mode [✓ Sync] Async
Updates (in ms) 100
Steps between updates 1

Figure 6.16: The window for setting mode and time parameters of the animation.

The "Animation" menu and the group of buttons on the toolbar, shown in Fig. 6.17, provide functions for control of the simulation process – start and stop of the simulation, setting the time to the initial time point (zero) and performing single simulation steps.

The animation process has been realized as follows. Once the command to start the animation has been given, the main loop is executed. This loop makes one call to simultaneously update all simulated components and generate the

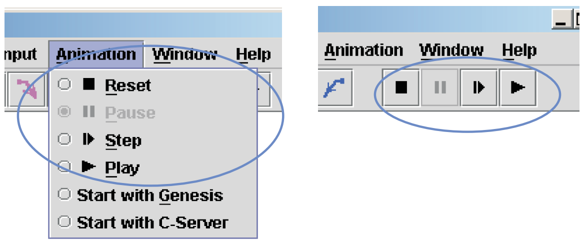


Figure 6.17: Menu and toolbar elements for animation control.

update results on the screen. In our simulation, spike events and neuron activity changes are shown for each simulation step.

The results of a simulation are saved as ordered vectors that are priority queues, a data structure that stores elements according to their priorities or "key" values [65]. In our simulation, the time when the event occurred is represented by the "key" value; the elements of the ordered vector are objects with two fields containing the parameter and time values. Elements can be accessed via the "key" value, i.e. at the actual time.

6.5.2 Graphical windows

The graphical windows (Fig. 6.18, Fig. 6.20) allow one to observe the curves representing the dynamics of simulated parameters. They are activated from the "Window" menu (Fig. 6.19) and represent additional information about the model structure in the main window. There are three kinds of windows showing information in graphical form: windows with membrane voltage curves, windows with channel conductance curves, and windows for spike event diagrams.

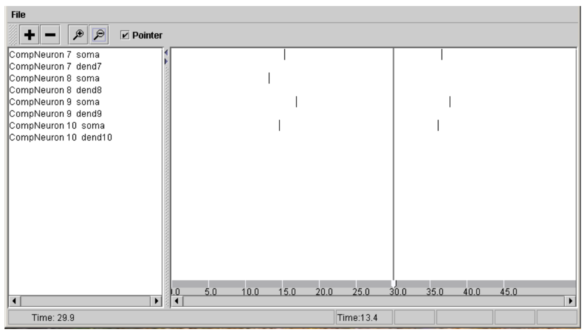


Figure 6.18: Graphical window of NeuroSim showing spike events.

Each graphical window contains two scrolling panels: the left panel shows a list of neural elements, whose properties are viewed in the right panel. The right panel is automatically updated after selecting elements to be represented.

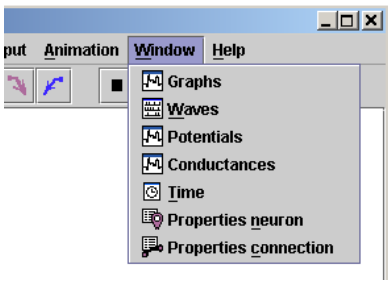


Figure 6.19: NeuroSim "Window" menu.

The panel with graphical information provides the possibility to set the simulation time (actual time) to any time index and run the simulation from that moment. It can be manipulated by using a time pointer existing in the right panel.

The timeline consists of the time pointer and the simulation time scale. The pointer's position can be shifted with the mouse to a new position. It will change the actual time of the simulation process, and the status of model elements in the main panel will be updated for this time index. Using buttons from the "Animation"menu, one can start the simulation process from a preset time index.

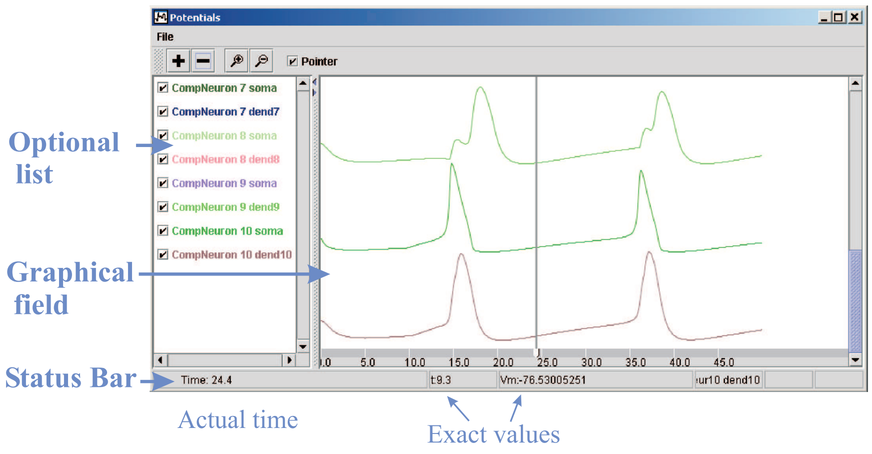


Figure 6.20: Graphical window of NeuroSim with membrane voltage curves.

The animation of spike events and neuron states shown in the main window is synchronized with the position of the time pointer showing the actual time of the simulation in a graphical window. Owing to this, one can view the state of a neural network and the dynamics of certain parameters corresponding to this time index. It makes the presentation of the results more effective.

While the animation process is in progress, the position of the time pointer is also modified. In the main window, one can view the "curve spread" and, at the same time, the neuron states corresponding to this time index. This feature gives for the user the possibility to more effectively analyze results.

The graph representation can be zoomed in/out in the time scale using the buttons located at the top of the graphical window.

The status bar at the bottom of all graphical windows shows the actual time of the simulation. It also can be used to read the values of the parameters and the corresponding time of the simulated curve.

6.6 Applet and application versions of NeuroSim

NeuroSim was realized in two versions: an applet version and an application version [42]. The application version can be run on a user's computer with various operating systems.

The applet version can be started from the Internet. As it was said above, the model description files and the files with descriptions of the GUI element states can be saved on the harddisk, even when the program started as an applet. Usually, Java applets are prevented from writing files to the hard disk and executing external programs. Therefore, the applet was signed using Java's `jarsigner` program to allow the applet to work with local resources.

If the browser supports the latest version of the Java Development Kit (at least JDK 1.3), the applet can be run with the browser's default virtual machine. By applet starting, the user needs to accept the signed applet properties shown in the dialog window. When the JDK is an older version, one needs to install the Java Plug-In, which allows browsers to run the most current version of the Java Runtime Environment (JRE). The applet was converted with Sun's HTML converter, to invoke the Java Plug-In on standard browsers. To start the applet successfully, the system's file policy should be configured using the certificate file, which grants an applet use of local system resources.

6.7 Summary

This chapter gives the researcher a description of model construction and simulation with NeuroSim. Since the model definition and visual presentation of simulation results are provided by the client's GUI, no programming skills are necessary from the modeller.

Through this chapter, the user gets an insight about the most important aspects of working with NeuroSim. First, the functions of the elements in the main window have been described. After that, the most important operations for model definition have been mentioned. Moreover, the processes of model definition and simulation setting has been described in the commonly occurring examples: the two-cell neural network with Hodgkin-Huxley channels, and the neuron containing the passive dendritic tree. The possible formats in which the model description files can be saved have been presented. XML data contains the model element properties in a hierarchial structure and is very important for the data exchange. The state of the models can be saved as serializable objects, since all neural elements were realized with serializable functionality.

Next, information on how the connection with the server can be established. The results of simulation can be visually presented using the graphical windows. Also, the data from the server can be played back: it was shown how the animations of neuron activations and pulse propagation through the network connections can be started.