

CONTENTS

CONTENTS	2
1. INTRODUCTION	4
2. WINDOWS INSTALLATION	5
2.1 INTRODUCTION.....	5
2.2 INSTALLATION OF MYSQL SERVER	5
2.3 SPIKESTREAM INSTALLATION.....	9
3. LINUX INSTALLATION.....	10
3.1 INTRODUCTION.....	10
3.2 INSTALLATION AND CONFIGURATION OF MYSQL SERVER	10
3.3 SPIKESTREAM INSTALLATION.....	11
4. SPIKESTREAM DATABASE CONFIGURATION TOOL	12
4.1 INTRODUCTION.....	12
4.2 CONFIGURING SPIKESTREAM DATABASES	12
5. ARCHITECTURE	14
5.1 DATABASES	14
5.2 NETWORKS.....	14
5.3 ARCHIVES.....	14
5.4 ANALYSES.....	14
5.5 RELATIONSHIP BETWEEN NETWORKS, ARCHIVES AND ANALYSES.....	14
6. CORE FUNCTIONALITY	16
6.1 NETWORKS TAB	16
6.2 3D NETWORK VIEWER.....	17
6.3 EDITOR TAB.....	17
6.4 VIEWER TAB	18
6.5 SIMULATION TAB.....	20
6.6 ARCHIVES TAB.....	20
6.7 ANALYSIS TAB	21
7. PLUGINS	22
7.1 INTRODUCTION.....	22
7.2 NETWORK PLUGINS.....	22
7.2.1 Aleksander Networks Builder	22
7.2.2 Tononi Networks Builder	23
7.2.3 Aleksander/Gamez Test Networks 2	23
7.3 NEURON GROUP PLUGINS.....	24
7.3.1 Cuboid Neuron Group Builder.....	24
7.4 CONNECTION GROUP PLUGINS.....	24
7.4.1 Random1 Connection Group Builder	24
7.5 SIMULATION PLUGINS.....	25
7.5.1 Nemo CUDA Simulator	25
7.6 ANALYSIS PLUGINS.....	26
7.6.1 Liveliness Analyzer.....	26
7.6.2 State-based Phi Analyzer.....	28
7.7 WRITING SPIKESTREAM PLUGINS.....	30
8. IMPORTING FROM NRM	31
8.1 INTRODUCTION.....	31
8.2 CREATING FILES IN NRM	31
8.3 IMPORTING NRM FILES INTO SPIKESTREAM	33

9. DATABASES	35
10. CONFIGURATION	36
10.1 INTRODUCTION	36
10.2 DATABASE SETTINGS	36
10.3 OTHER SETTINGS	36
11. KEYBOARD SHORTCUTS	37
NETWORK VIEWER NAVIGATION	37
OTHER SHORTCUTS.....	37
REFERENCES	38

1. INTRODUCTION

The current version of SpikeStream is a network creation and analysis tool. The next version will support CUDA hardware accelerated simulation of neural networks using Nemo (<http://www.doc.ic.ac.uk/~akf/nemo/index.html>), which will be released in the third quarter of 2010.

This manual covers most of the basic features of SpikeStream. If you have any questions, feel free to use the SpikeStream mailing list (spikestream-user@lists.sourceforge.net) or contact me, David Gamez, at: <http://www.davidgamez.eu/pages/contact/index.php>.

NOTE: This manual is work in progress and some sections are incomplete or missing.

NOTE: I have not had time to fully test SpikeStream, so it is likely to crash or show inconsistent behaviour from time to time. If you let me know the exact steps that lead to each crash, I will fix them ASAP.

2. WINDOWS INSTALLATION

2.1 Introduction

This section takes you through the minimum steps that are needed to get SpikeStream running on Windows. The installation has the following stages:

1. Install and configure a MySQL server to hold the SpikeStream databases.
2. Install the SpikeStream application.
3. Configure SpikeStream so that it can communicate with the MySQL databases.

2.2 Installation of MySQL Server

The SpikeStream data is stored on three MySQL databases and SpikeStream will not run unless it can communicate with these. This section describes how to get a MySQL server running on your local machine. You can skip this section if you already have a MySQL server on a local or remote host that you want to use with SpikeStream.

The first step is to download the MySQL community server from: <http://dev.mysql.com/downloads/mysql/>. The Windows (x86, 32-bit), MSI Installer works fine, although other versions should be ok as well. Save the installer to disk and double click on it to run it. Go through the installation steps using the default options and on the last screen you will be offered the chance to configure the server as shown in Figure 1.



Figure 1. Last stage of MySQL installation dialog. Select “Configure the MySQL Server now” and click on “Finish” to launch the MySQL Server Instance Config Wizard. Registration of the MySQL server is unnecessary.

If you accidentally click past this step, you can launch the “MySQL Server Instance Config Wizard” by clicking **Start->Programs->MySQL->MySQL Server 5.1->MySQL Server Instance Config Wizard** after the MySQL server has been installed.

The MySQL Server Instance Configuration Wizard should launch as shown in Figure 2.

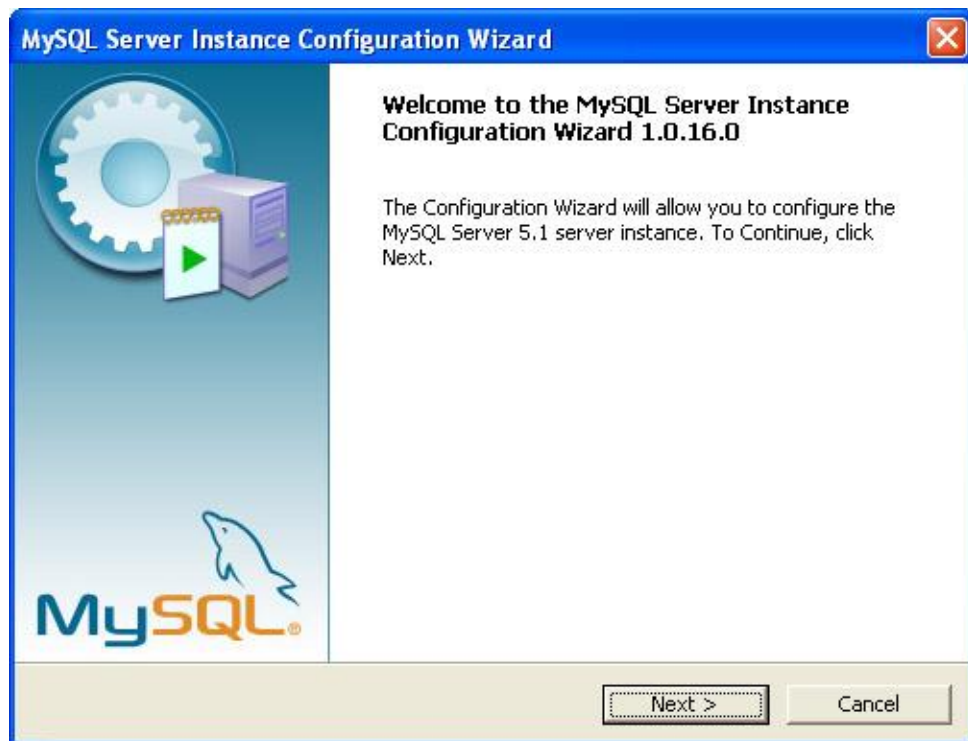


Figure 2. MySQL Server Instance Configuration Wizard

Click on “Next” and you will be presented with the dialog shown in Figure 3.

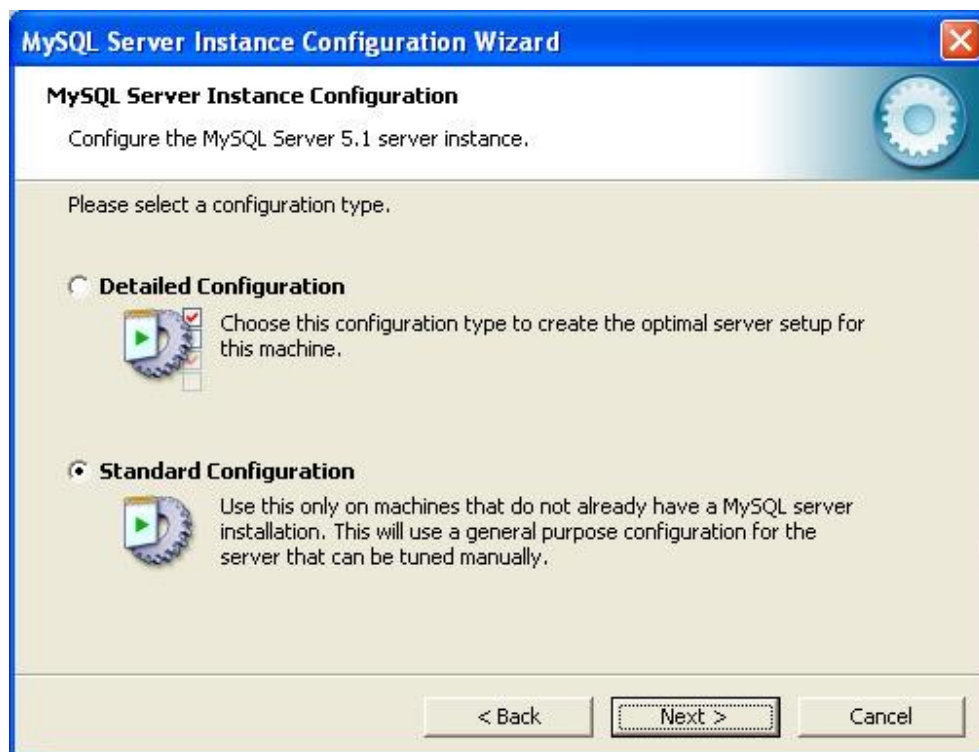


Figure 3. Configuration type. Select “Standard Configuration” unless you are using MySQL for other applications and want to configure it differently. How the server is configured does not currently make much difference to SpikeStream, but the Standard Configuration is easier to set up.

Select “Standard Configuration” and click “Next”. You will be presented with the dialog shown in Figure 4.



Figure 4. Install MySQL as a Windows service. SpikeStream does not need the MySQL bin directory to be included in the Windows PATH, but it will not create any problems if you select this option.

Select “Install as Windows Service” and click “Next”. This will show you the dialog in Figure 5.



Figure 5. MySQL security settings. This is the most critical part of the installation because you need to remember the password that you enter.

Select “Modify Security Settings” and enter a password for the root account on the MySQL server. *Make sure that you remember the password because you will have to provide this information to the SpikeStream Database Configuration Tool* (see Section 4). Note that SpikeStream does not need to be configured with the root account and

it is possible to run each database using a different host, user and password. When you have written down or memorized the root password, click “Next” and you will be shown the dialog in Figure 6.

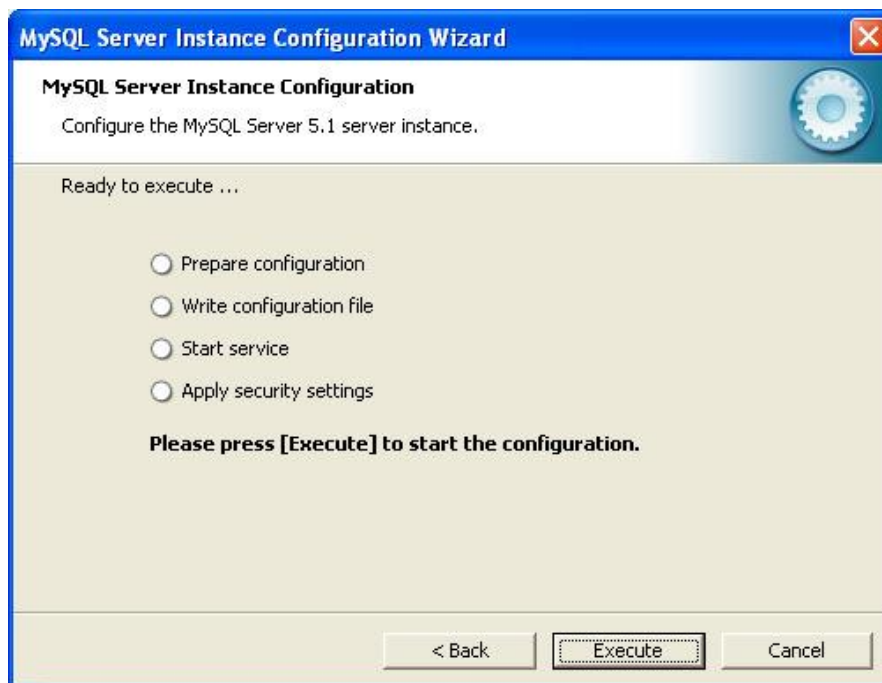


Figure 6. Execution dialog.

Click on “Execute” to save your settings. If the operation is successful you should see the dialog shown in Figure 7.

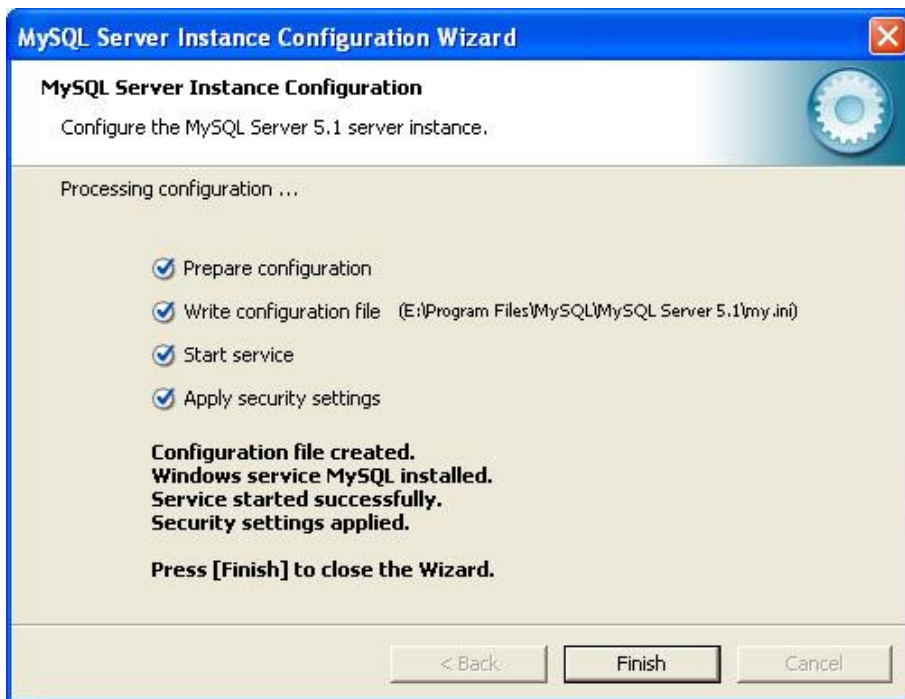


Figure 7. MySQL successful configuration.

You should now have a MySQL server running on your local machine whose root access has been configured using the password that you provided. The next step is to add the SpikeStream databases to this MySQL server and configure SpikeStream with the correct user name and password so that it can access this server. These steps can be carried out manually (see Section 9) or using the SpikeStream Database Configuration Tool (see Section 4). Before you can use this tool, you need to download and install SpikeStream.

2.3 SpikeStream Installation

NOTE: A proper installer will be created soon.

The Windows version of SpikeStream is currently distributed as a zipped file, `spikestream-0.2.zip`. Download this file and unzip it to the place where you want to install SpikeStream. When you open up the folder `spikestream-0.2` you should see the files shown in Figure 8.

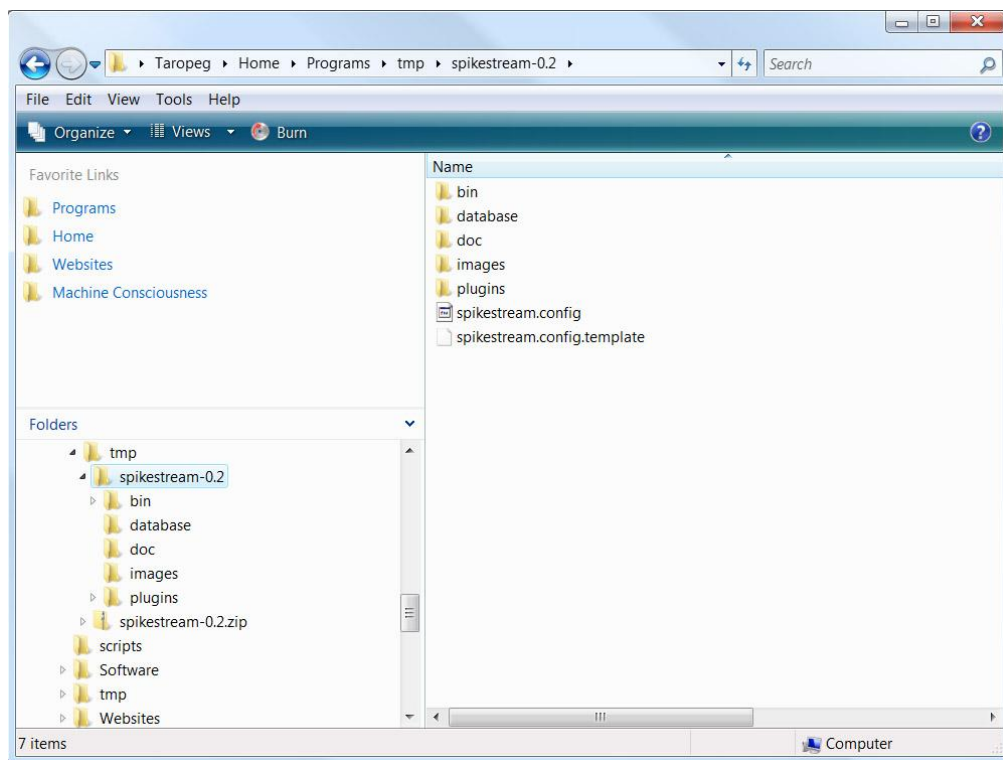


Figure 8. SpikeStream files shown in Windows Explorer.

SpikeStream is run by double clicking the “`spikestream.exe`” file in the “`bin`” directory. It will not work until the databases have been configured using the SpikeStream Database Configuration Tool, which is described in Section 4.

3. LINUX INSTALLATION

NOTE: SpikeStream should build and run on Linux and OS X. However, building and running SpikeStream has only been tested on Windows so far.

3.1 Introduction

Before SpikeStream can run, the correct databases need to be created and their host, username and password entered in the SpikeStream Database Configuration Tool.

3.2 Installation and Configuration of MySQL Server

MySQL typically forms part of a Linux distribution and it is also available at www.mysql.org. You will need to install the development parts of MySQL as well as the server.

When you have installed MySQL, test to see if it is running using: `ps -el | grep mysql`. This should return a line containing “mysqld” as one of the running processes. If this is not listed, use `chkconfig` to enable the service. As superuser type: `chkconfig --list mysql`, which should tell you if mysql is enabled or not. If it is not enabled for your current run level, type: `chkconfig mysql on` and make sure that it is enabled.

Even when mysql is enabled, the daemon may not have started. To start the daemon go to `/etc/init.d/` and log in as root. Then run the mysql command by typing: `./mysql start`, which should start up the daemon. Check that it has started, then you are ready to set up the accounts.

You need to allow external access to MySQL if you are running SpikeStream on a different machine to the database server, and your system's firewall may need to be changed to facilitate this. In SUSE this can be done by adding MySQL to the firewall configuration using YAST.

When MySQL is installed it typically comes with one unsecure root account. It is recommended that you secure the root account and create a separate account for SpikeStream. The following steps should work:

- Log in to MySQL as root using `mysql -u root`
- Display the current accounts: `SELECT user, host, password FROM mysql.user;`
- Set a password for root: `SET password=PASSWORD("secretpassword")`
- Get rid of unnecessary users: `DELETE FROM mysql.user WHERE user != "root";`
- Get rid of logins from outside machine: `DELETE FROM mysql.user WHERE host != "localhost";`
- Create accounts with the user ‘SpikeStream’ and the password ‘myPassword’ that can access the database on localhost or a subnetwork:
 - `GRANT ALL ON *.* TO SpikeStream@localhost IDENTIFIED BY "myPassword";`
 - `GRANT ALL ON *.* TO SpikeStream@"192.168.1.0/255.255.255.0" IDENTIFIED BY "myPassword";`
- If these accounts have been created successfully it should be possible to log into the database locally or from another machine on the same network using:
 - `mysql -uSpikeStream -pmyPassword` (local login with password “myPassword”)
 - `mysql -uSpikeStream -pmyPassword -h192.168.1.9` (remote login with mysql hosted on 192.168.1.9 and password “myPassword”)

You can create a different account for each database and the databases can be hosted on different machines. You need to remember the details for each database and enter them into the SpikeStream Database Configuration Tool. These details can also be manually into the `spikestream.config` file (see Section 10).

3.3 SpikeStream Installation

At the time of writing the Linux installation of SpikeStream has not been fully tested. SpikeStream is based on Qt and to build it you will need to have a recent version of Qt (4.5 or greater) installed on your system. SpikeStream also depends on GMP, which forms part of most Linux distributions, and qwt, which is available at: <http://qwt.sourceforge.net/>.

The basic steps for installing on Linux are:

- Check out the latest version of the SpikeStream source code using SVN: `svn co https://spikestream.svn.sourceforge.net/svnroot/spikestream spikestream`
- Change to the root directory of SpikeStream: `cd spikestream/trunk`
- Check all of the .pro files to make sure that they point to the correct include and library locations. This inconvenience will be fixed in the next release!
- Run qmake to generate the Makefile: `qmake spikestream.pro` (you may need to specify the complete path to qmake if you have installed it yourself).
- Build SpikeStream by typing `make`.

When SpikeStream has built, you will need to set the paths as follows for it to run:

- `SPIKESTREAM_ROOT=/home/testuser/projects/spikestream/trunk`
- `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/qwt/lib:$SPIKESTREAM_ROOT/lib:$SPIKESTREAM_ROOT/plugins/analysis`

Before SpikeStream can run it is necessary to set up the SpikeStream databases. This can be done manually, or using the SpikeStream Database Configuration Tool, which is described next.

4. SPIKESTREAM DATABASE CONFIGURATION TOOL

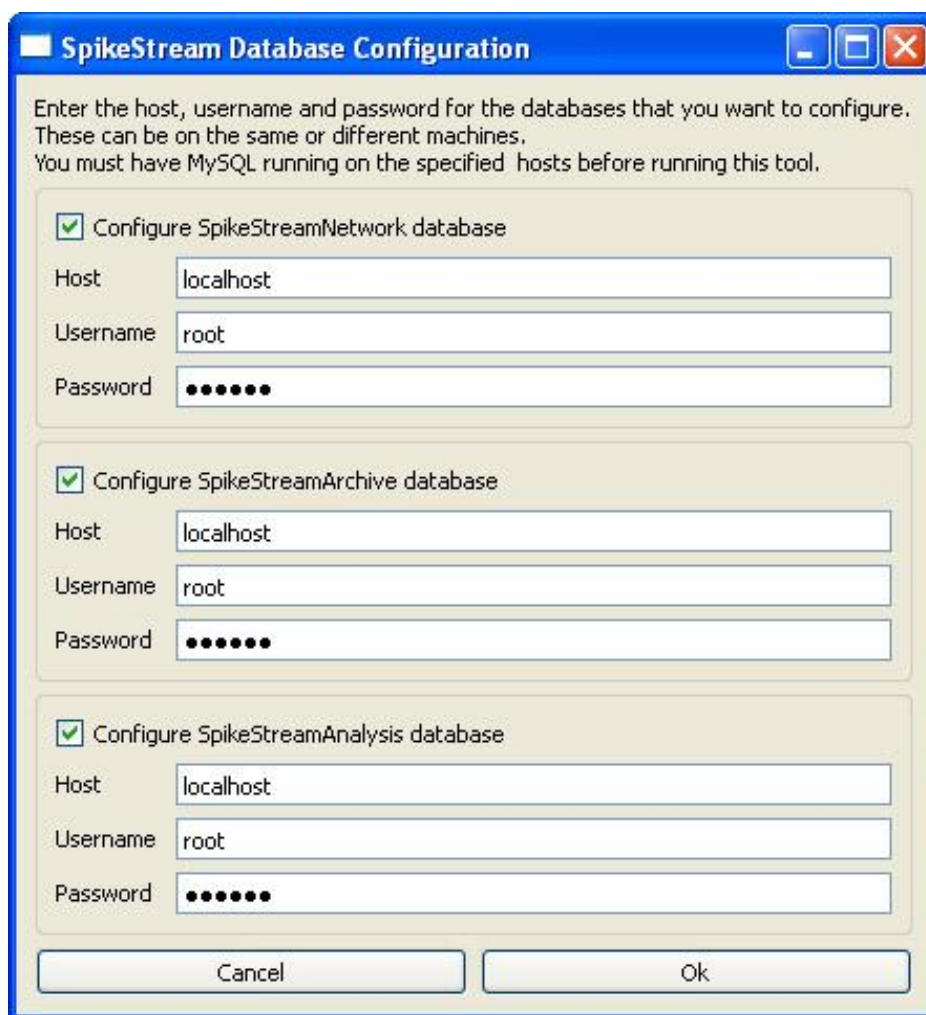
4.1 Introduction

This tool creates the databases for SpikeStream and stores their settings in the SpikeStream configuration file (see Section 10). Before launching this tool you need to have the details of the MySQL server that will be hosting the SpikeStream databases. If you are running MySQL on a local machine, then you will just need the username (possibly “root”) and password of an account that has permission to create databases.

4.2 Configuring SpikeStream Databases

WARNING: This tool completely resets the selected SpikeStream databases. Any data that is currently on these databases will be lost.

To launch the database configuration tool, open the “bin” directory in the SpikeStream installation and double click on the file “dbconfigtool.exe”. This launches the application shown in Figure 9.



SpikeStream Database Configuration

Enter the host, username and password for the databases that you want to configure. These can be on the same or different machines. You must have MySQL running on the specified hosts before running this tool.

☒ Configure SpikeStreamNetwork database

Host: localhost

Username: root

Password:

☒ Configure SpikeStreamArchive database

Host: localhost

Username: root

Password:

☒ Configure SpikeStreamAnalysis database

Host: localhost

Username: root

Password:

Cancel Ok

Figure 9. SpikeStream Database Configuration Tool. All three databases have been selected for configuration and the host, username and password entered for each. The host, username and password can be different for each database and it is possible to configure just one or two of the databases.

Click on the check box for the databases that you want to configure and enter the requested information. The settings for each database can be the same or different and only the selected database(s) will be configured. Warning dialogs will be shown if the MySQL server(s) cannot be found or if the databases already exist. When you have entered the requested information you will be presented with the dialog shown in Figure 10.

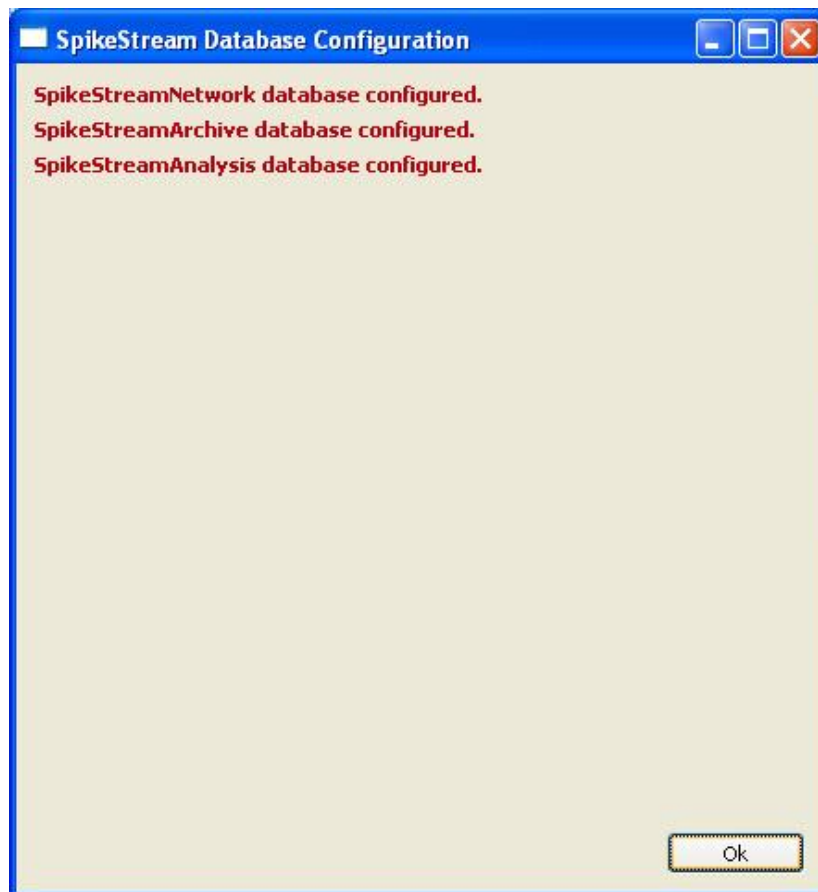


Figure 10. Results page of SpikeStream Database Configuration Tool. These results will vary depending on the databases that have been selected for configuration.

It should now be possible to run SpikeStream by double clicking on "spikestream.exe" in the bin directory of the SpikeStream installation.

WARNING: When data already exists in the SpikeStream databases it is recommended to reconfigure all three databases at once. Otherwise there are likely to be data inconsistencies.

5. ARCHITECTURE

5.1 Databases

SpikeStream is based around three databases that hold all of its data. This means that pretty much everything that you see within SpikeStream has already been saved, so there is no need to save a network or analysis at the end of a session. If SpikeStream crashes, you will generally not lose any data, but it is possible that the databases will be left in an inconsistent state if SpikeStream crashes in the middle of an operation.

WARNING: If you wipe the database - for example, using the MySQL client, the database configuration tool or from within SpikeStream - then you lose everything, unless you have backed it up.

There is currently no way within SpikeStream to back up your data to an external file and to restore it at a later point in time. The best way to do this is using the mysqldump program, which is described in the following articles:

- <http://dev.mysql.com/doc/refman/5.1/en/mysqldump.html>
- <http://www.devarticles.com/c/a/MySQL/Backing-Up-Your-MySQL-Databases-With-MySQLDump/>

It is *strongly recommended* that you practice backing up and restoring your data with mysqldump before using it on important data. More information about the SpikeStream databases is given in Section 9.

5.2 Networks

The current networks are listed on the Networks Tab, which is covered in Section 6.1. A network contains information about the neurons and the connections between them. Information about networks is stored in the SpikeStreamNetwork database.

5.3 Archives

Each archive holds a sequence of one or more time steps containing the firing patterns of the neurons in the network at that time step. Archives can be viewed and played back on the Archives Tab, which is described in Section 6.6. Archives are stored in the SpikeStreamArchive database.

5.4 Analyses

Analyses in SpikeStream are handled as dynamically loaded plugins, which are managed using the Analysis Tab described in Section 6.7. Analyses are stored in the SpikeStreamAnalyses database.

5.5 Relationship Between Networks, Archives and Analyses

The structure of the data within SpikeStream mirrors that of the databases. At the top of the hierarchy is a network and only one network can be loaded into SpikeStream at a time. Each network is associated with one or more archives that contain the firing patterns of the network, and each firing pattern is associated with one or more analyses. This hierarchy is shown in Figure 11.

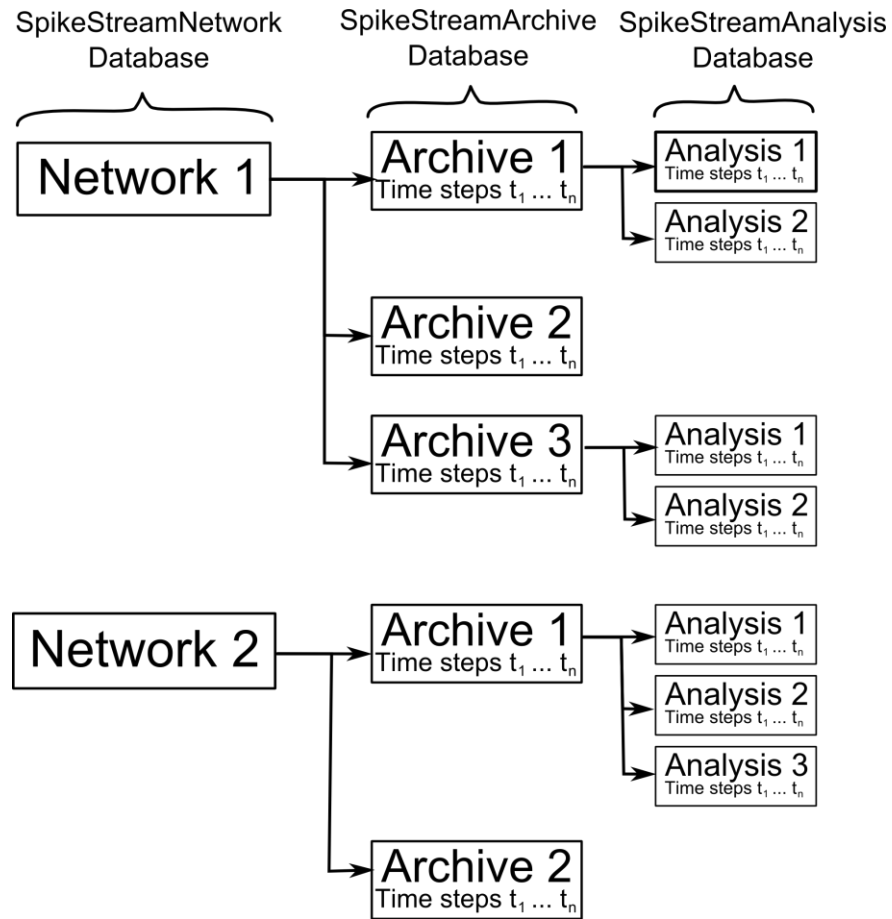


Figure 11. Relationship between networks, archives and analyses.

Each archive is an archive of a particular network and each analysis is an analysis of a particular state of a particular network.

WARNING: An archive is meaningless without its associated network, and so an archive will be deleted if its network is deleted. Similarly, an analysis is meaningless without its associated archive and it will be deleted if this archive is deleted.

6. CORE FUNCTIONALITY

6.1 Networks Tab

When you launch SpikeStream you are presented with the Networks Tab, as shown in Figure 12.

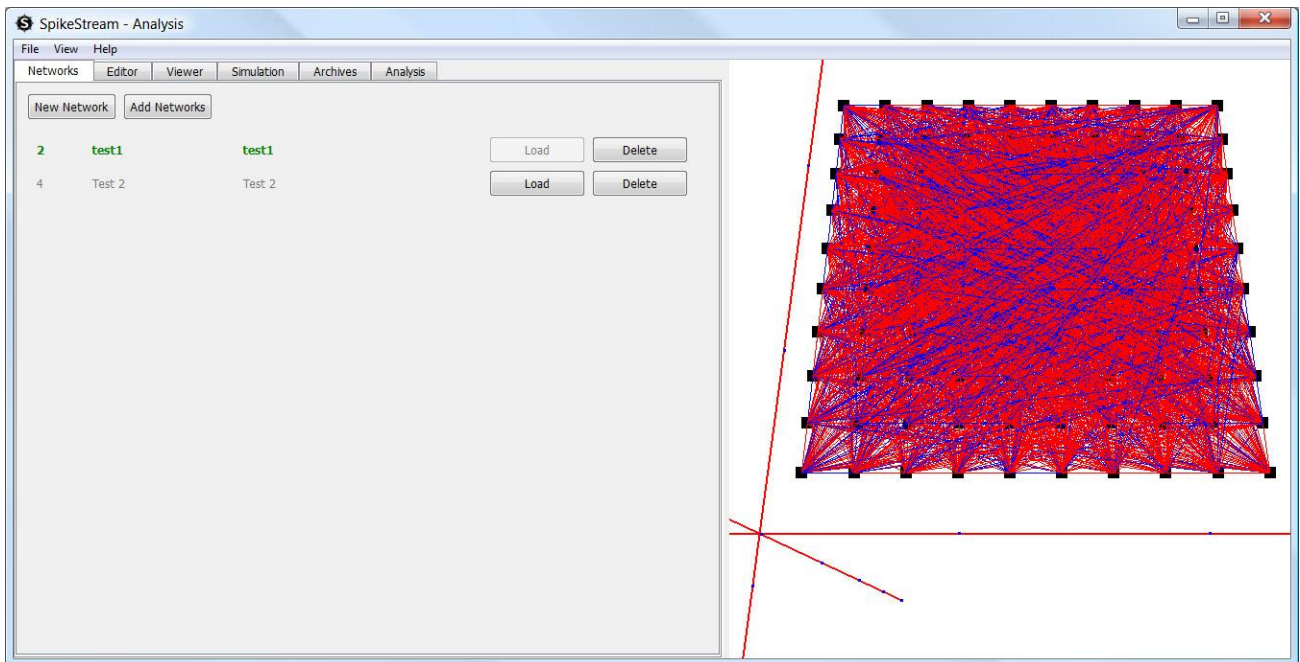


Figure 12. Networks Tab

Clicking on “New Network” launches a dialog that enables you to enter a name and description of the new network. Enter the name and description and click “Ok” to create the new network.

Clicking on “Add Networks” launches the Add Networks dialog shown in Figure 13.

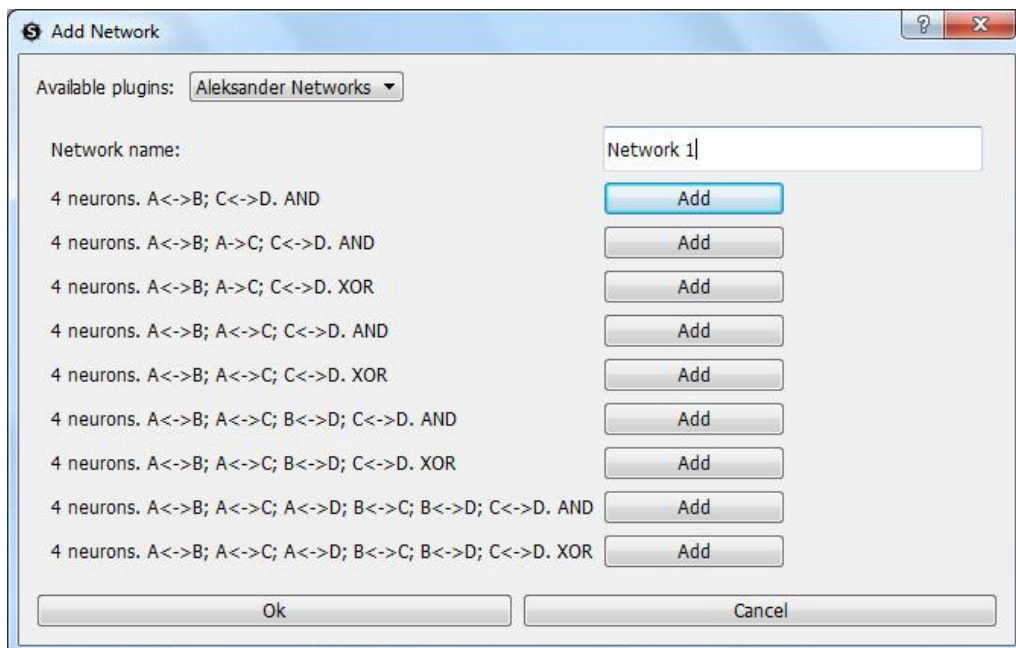


Figure 13. Add Networks dialog showing the Aleksander Network plugin (see Section 7.2.1)

The Add Networks dialog displays the currently available network plugins. This type of plugin is designed to add entire networks to the SpikeStream database, and possibly associated archives as well. When more than one

network plugin is available you can select between network plugins using the combo box at the top of the dialog. The SpikeStream distribution currently includes three network plugins, which are documented in Section 7.2.

The networks in the SpikeStream database are listed in the Networks Tab, as shown in Figure 12. Clicking on “Load” will load the network into SpikeStream; clicking on “Delete” removes the network and its associated archives and analyses from the SpikeStream databases.

WARNING: Clicking on delete will permanently remove the network and all of its associated archives and analyses from the SpikeStream database. This step cannot be undone.

The shortcut **F1** can be used to switch to the Networks Tab.

6.2 3D Network Viewer

The right side of SpikeStream is taken up with the 3D Network Viewer, which displays the current network in three dimensions. The 3D Network Viewer has the following features:

- Navigation in the 3D Network Viewer is carried out using the shortcuts described in Section 10.
- Neuron and connection groups can be shown or hidden in the 3D Network Viewer using the Editor tab, as described in Section 6.3.
- Double clicking on a neuron in the 3D Network Viewer enables you to see the properties of its connections in the Viewer Tab, as described in Section 6.4.
- When archives are played back neurons are highlighted in the 3D Network Viewer as described in Section 6.6.
- The results of analyses can be displayed in the 3D Network Viewer, as described in Section 7.6.

In its initial state, the 3D Network Viewer is organized with the positive Z axis pointing upwards, the positive Y axis pointing away from the viewer and the positive X axis moving left to right.

6.3 Editor Tab

The Editor Tab shown in Figure 14 is used for editing an existing network. Clicking “Add Neurons” displays a dialog showing the available plugins for adding neuron groups. Clicking “Add Connections” displays a dialog showing the available plugins for adding connection groups. The plugins for adding neuron or connection groups are described in Section 7.3 and Section 7.4.

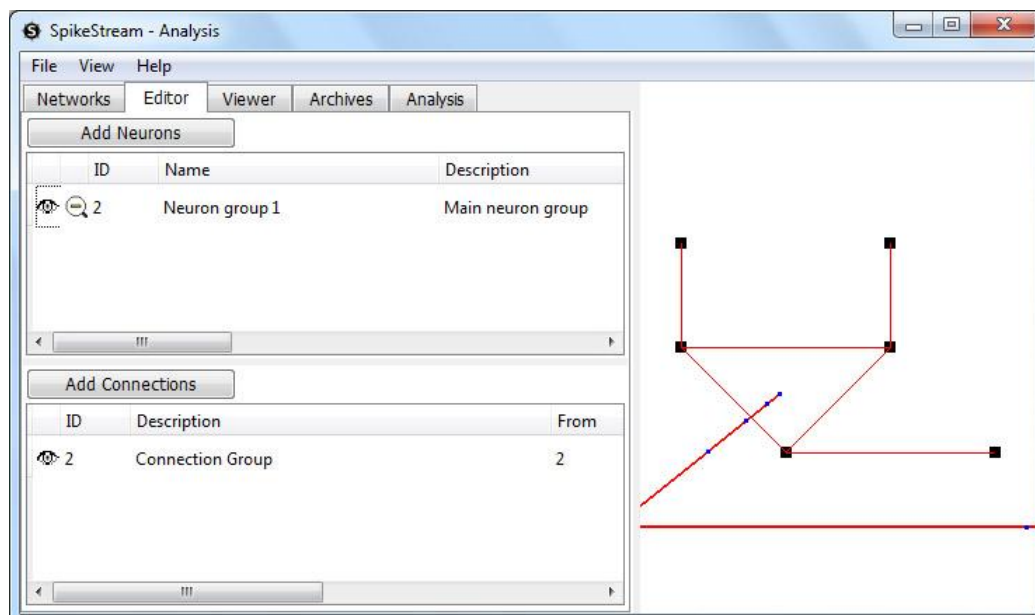


Figure 14. Editor Tab

A network is composed of one or more neuron and connection groups and these can be shown or hidden by clicking on the eye icon in the table listing the neuron or connection groups. The neuron group table also has a magnifying glass icon. A single click on this icon zooms in to the side of the corresponding neuron group; a second click zooms above the corresponding neuron group; a third click zooms out to view the entire network.

The shortcut **F2** can be used to switch to the Editor Tab.

6.4 Viewer Tab

The Viewer Tab is used to set properties of the 3D Network Viewer – see Section 6.2 – and to display details about the connections TO or FROM a selected neuron.

To view the connections TO and/or FROM an individual neuron, double click on the neuron in the 3D Network Viewer. When the neuron is selected it turns bright green and only the connections TO and/or FROM the neuron are shown – see Figure 15. This feature is at an early stage of development and double clicking to select neurons is easier when zoomed out from the network. Note that only connections set to visible in the Editor Tab are shown, although all connections are listed in the table.

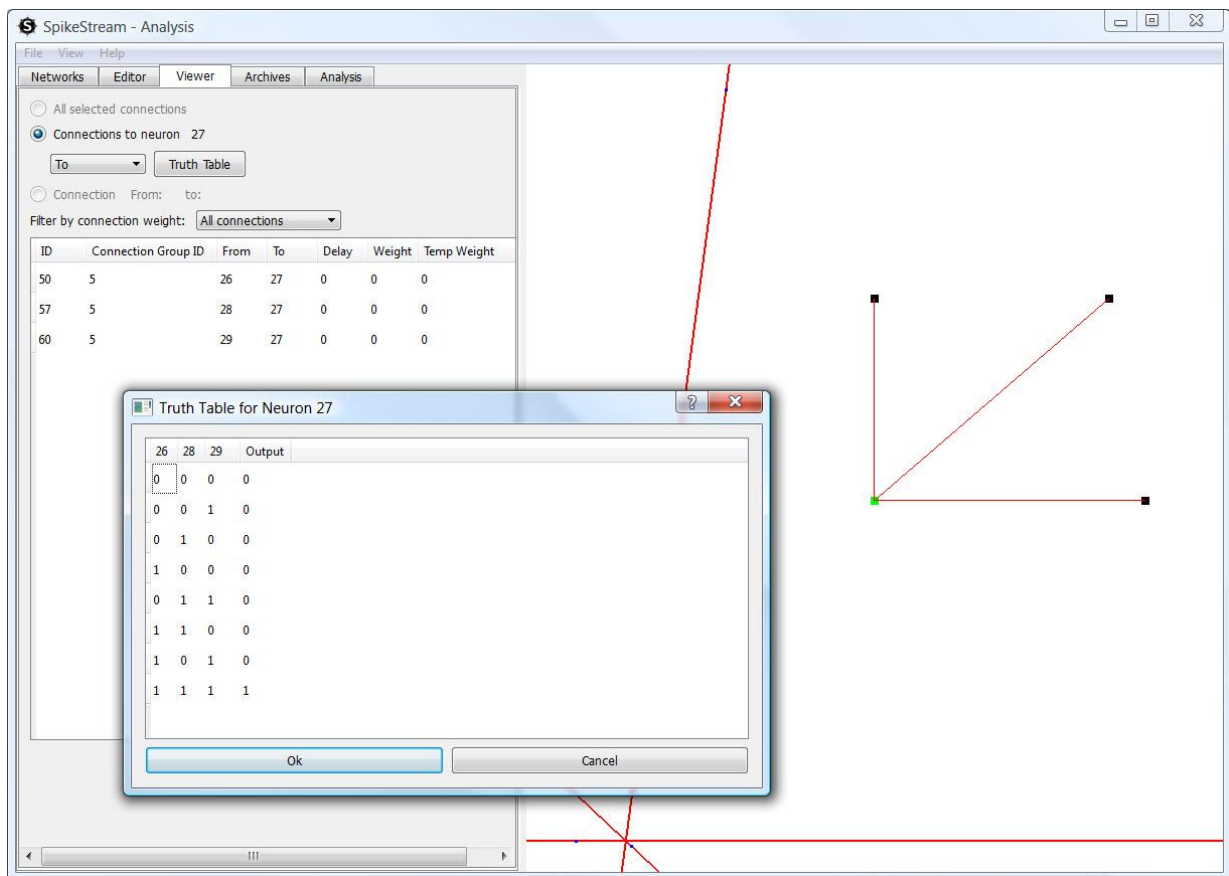


Figure 15. Viewer Tab showing connections TO a weightless neuron that was selected by double clicking, and a dialog displaying the selected neuron's truth table

The From/To combo box can be used to select whether connections FROM and/or TO the selected neuron are shown and listed in the table. When just TO connections are shown, the truth table for a weightless neuron can be viewed by clicking the Truth Table button.

When one neuron is selected, the connections between that neuron and another neuron can be viewed by holding down CTRL and double clicking another neuron. The second neuron appears coloured purple and the connections FROM the first neuron TO the second neuron are shown in the Viewer Tab, as shown in Figure 16.

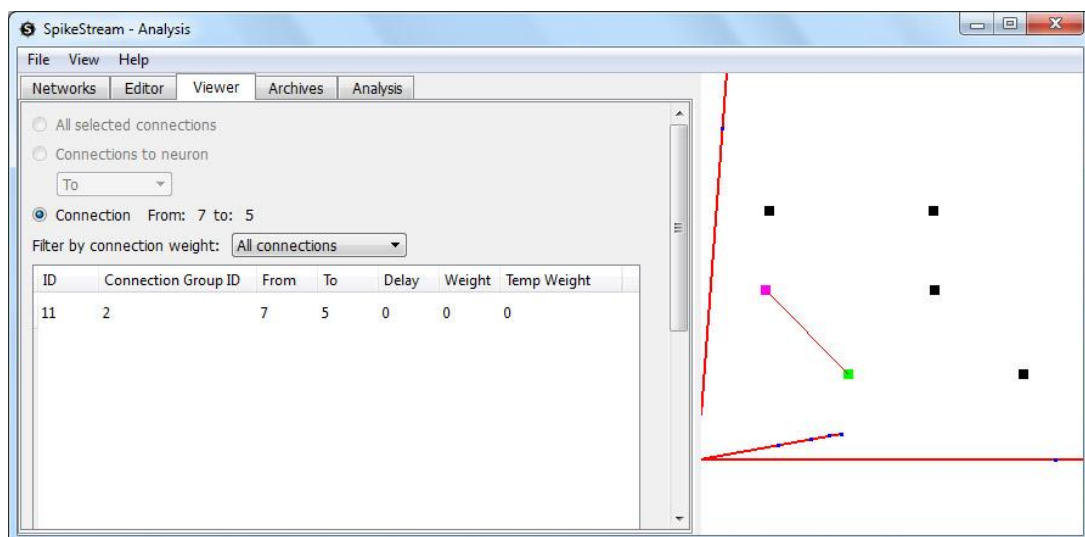


Figure 16. Viewer Tab showing connections from the neuron highlighted in green to the neuron highlighted in purple.

NOTE: The connection viewing mode takes precedence over the archive playing and analysis highlighting. You need to deselect a neuron that you have double clicked before playing back an archive or highlighting a cluster.

The shortcut **F3** can be used to switch to the Viewer Tab.

6.5 Simulation Tab

The Simulation Tab loads up and displays all of the available simulation plugins. When more than one plugin is available, the plugins can be selected using the combo box at the top of the Simulation Tab. The current simulation plugins are covered in Section 7.5.

6.6 Archives Tab

The Archives Tab, shown in Figure 17, is used to play back archived states of the network. Each archive consists of one or a number of time steps containing the neurons that were firing at that time step.

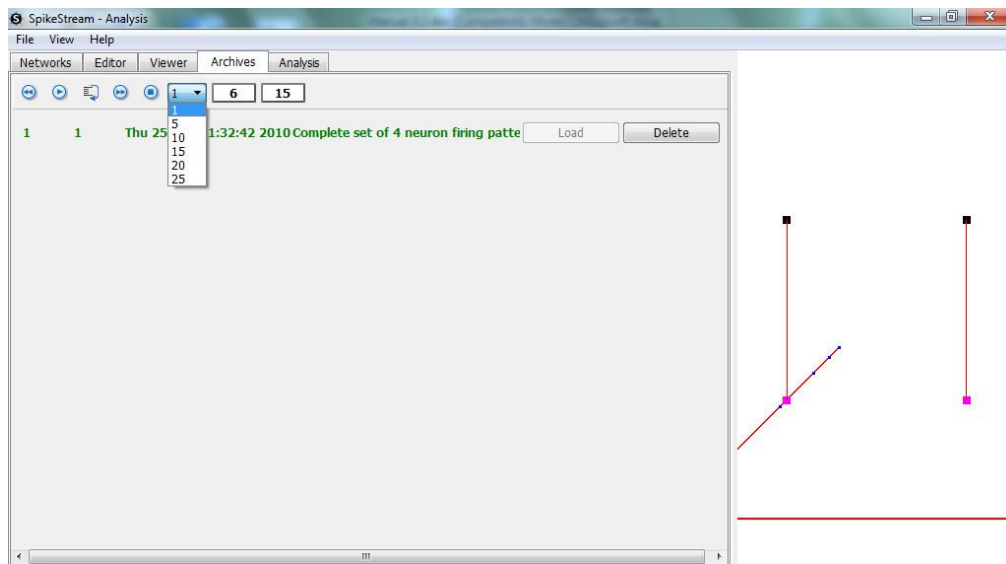







Figure 17. Archives tab

Each network can be associated with one or more archives. Clicking on the “Load” button causes the corresponding archive to be loaded up ready to play. Clicking on the “Delete” button permanently removes the archive and any associated analyses from the SpikeStream databases.

WARNING: Clicking on the delete button causes the archive to be permanently deleted from the database. This step cannot be undone.

When an archive is loaded the following controls can be used to play back the archive in the 3D Network Viewer:

-  Rewinds an archive back to the beginning.
-  Plays the archive. The combo box next to the stop button sets the speed of playback.
-  Steps through the archive one time step at a time.
-  Fast forwards through the archive.
-  Stops playback of the archive.

The shortcut **F4** can be used to switch to the Archives Tab

6.7 Analysis Tab

The Analysis Tab loads up and displays all of the available analysis plugins. When more than one plugin is available, the plugins can be selected using the combo box at the top of the Analysis Tab. The current distribution of SpikeStream includes plugins to carry out liveliness and state-based phi analyses, which are covered in Section 7.6.

7. PLUGINS

7.1 Introduction

Many of the functions of SpikeStream are organized using a plugin architecture, which make it easy to extend the functionality without altering the main code base. Plugins can use the `spikestream` and `spikestreamapplication` libraries or they can be entirely independent code. The minimal requirements for a SpikeStream plugin are described in Section 7.7.

SpikeStream currently supports the following types of plugin:

- **Network plugin.** Creates entire networks, including neurons, connections and possibly archives. Should be installed in the `plugins/networks` folder.
- **Neurons plugin.** Creates neuron groups within the currently loaded network. Should be installed in the `plugins/neurons` folder.
- **Connections plugin.** Creates connection groups within the currently loaded network. Should be installed in the `plugins/connections` folder.
- **Simulation plugin.** Carries out simulations and stores the result in the `SpikeStreamArchive` database. Should be installed in the `plugins/simulation` folder.
- **Analysis plugin.** Analyzes the currently loaded network. Should be installed in the `plugins/analysis` folder.

The next couple of sections cover the plugins that form part of the current SpikeStream distribution. The functionality of these plugins reflects the research interests of the current SpikeStream developers, and other people are encouraged to write and distribute their own plugins to extend SpikeStream's functionality. Instructions for writing plugins are given in Section 7.7.

7.2 Network Plugins

7.2.1 Aleksander Networks Builder

This plugin was written to add test networks for exploring ideas about information integration. It can be viewed by clicking on Add Networks on the Networks Tab and selecting "Aleksander Networks" from the drop down combo. This plugin is illustrated in Figure 13.

The names of each network indicates the connectivity and function of the neurons. For example, the third network in the list has the name "A<->B; A->C; C<->D. XOR", which describes the connectivity shown in Figure 18.

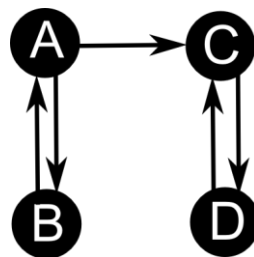


Figure 18. Aleksander plugin network: "A<->B; A->C; C<->D. XOR". Each of the neurons has an XOR function.

Clicking the "Add" next to the label adds the network to the database. If you are unsure about the connectivity and the functions of these networks, it is easy to add them and view their properties in the Viewer Tab, as described in Section 6.4.

The Aleksander Networks Builder also adds an archive with 16 time steps to the database for each network that contains all of the possible firing patterns of the four neurons. These firing patterns can be played back using the Archives Tab, which is described in Section 6.6.

7.2.2 Tononi Networks Builder

The Tononi Networks Builder plugin (see Figure 19) was written to add some of the networks described in Balduzzi and Tononi (2008). This plugin also adds an archive with a single time step that contains the firing pattern of the network as described in Balduzzi and Tononi (2008). Click on the “Add” button to add the network to the SpikeStream database.

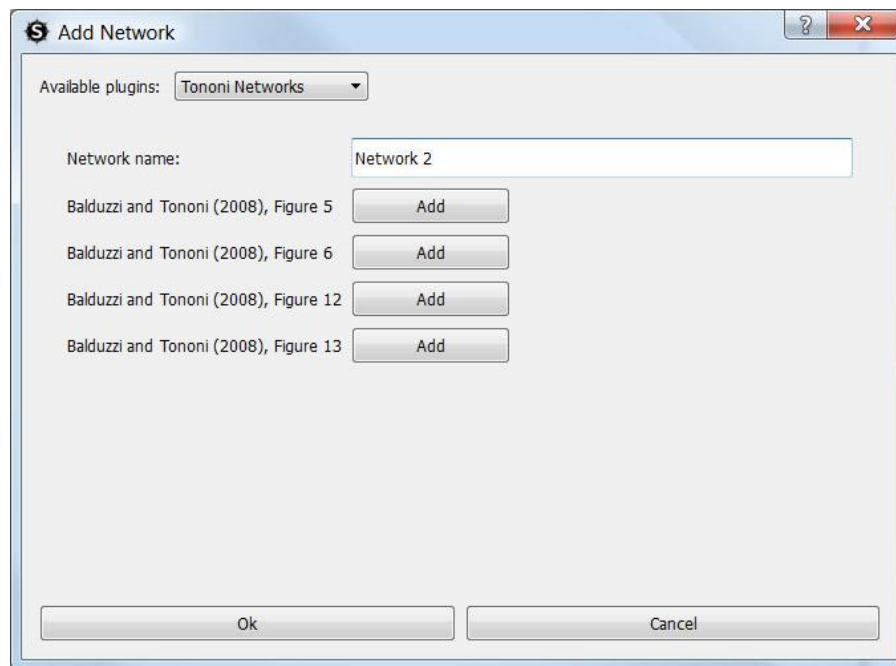


Figure 19. Tononi Networks Builder

7.2.3 Aleksander/Gamez Test Networks 2

This plugin (see Figure 20) adds a selection of 12 neuron networks that were used to compare the State-based ϕ measure of information integration with the liveliness measure. Click on the “Add” button to add the network to the SpikeStream database.

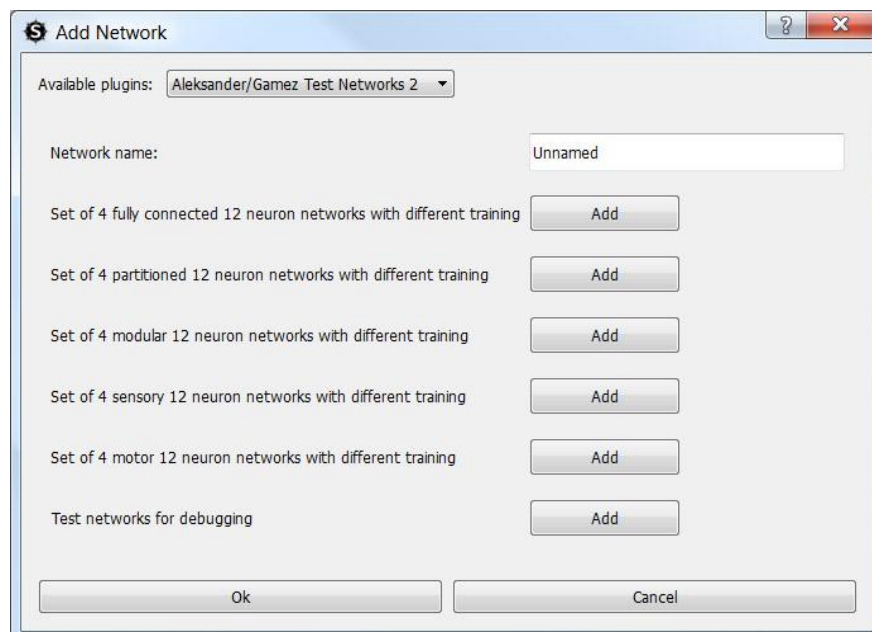
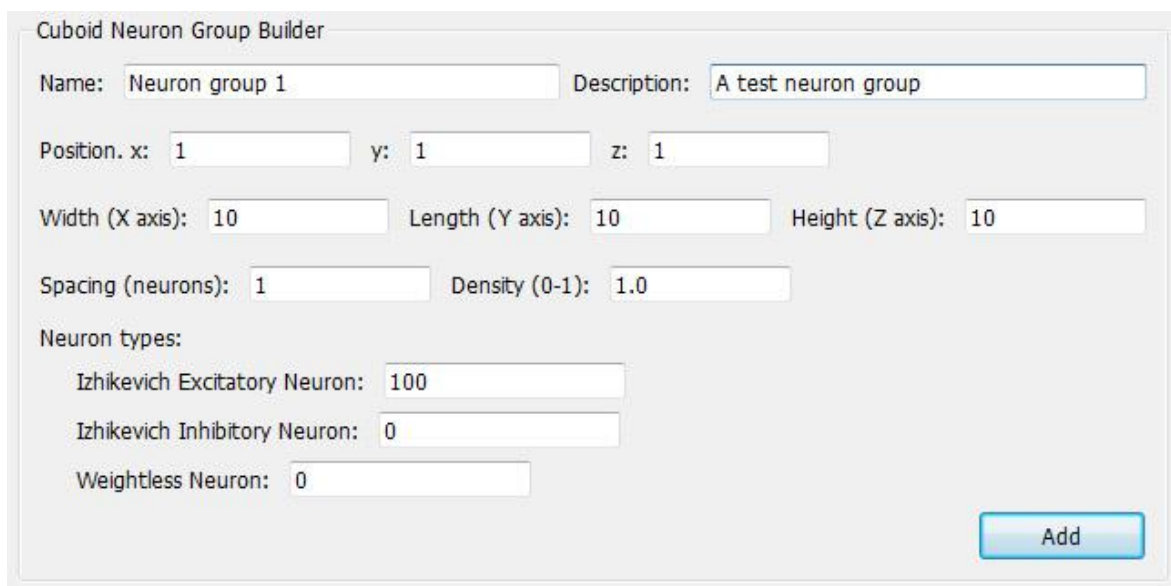


Figure 20. Aleksander/Gamez Test Networks 2 plugin

7.3 Neuron Group Plugins

7.3.1 Cuboid Neuron Group Builder

The Cuboid Neuron Group Builder plugin is shown in Figure 21. It enables the user to add a cuboid or planar neuron group to the network. When a number of different neuron types are included, these are added to the network as separate neuron groups.



The screenshot shows a window titled "Cuboid Neuron Group Builder". It contains several input fields for configuring a neuron group. The "Name" field is set to "Neuron group 1" and the "Description" field is set to "A test neuron group". The "Position" fields for x, y, and z are all set to 1. The "Width (X axis)", "Length (Y axis)", and "Height (Z axis)" fields are all set to 10. The "Spacing (neurons)" field is set to 1 and the "Density (0-1)" field is set to 1.0. Under the "Neuron types:" section, there are three rows: "Izhikevich Excitatory Neuron" set to 100, "Izhikevich Inhibitory Neuron" set to 0, and "Weightless Neuron" set to 0. An "Add" button is located in the bottom right corner.

Figure 21. Cuboid Neuron Group Builder plugin.

The parameters for this plugin are as follows:

- *Name*. The name of the neuron group.
- *Description*. A brief description of the neuron group.
- *Position*. The point on the neuron group closest to the origin. It is recommended to keep this value positive since it has not been tested with negative positions.
- *Width, length, height*. The width, length and height of the neuron group in neurons.
- *Spacing*. The spacing between neurons in the group.
- *Density*. The probability that a neuron will be created at a particular position.
- *Neuron types*. The proportion of each type of neuron that will be included in the cuboid. These numbers must add up to 100 and a separate neuron group will be created for each type.

7.4 Connection Group Plugins

7.4.1 Random1 Connection Group Builder

The Random1 Connection Group Builder plugin is shown in Figure 22. It enables the user to add random connections within or between neuron groups.

Random1 Connection Group Builder

Description: Undescribed

From: Neuron group 1 (3) To: Neuron group 1 (3)

Weight range 1 from: -1.0 to: 0.0 Proportion weight range 1: 20 %

Weight range 2 from: 0 to: 1.0

Delay (ms) from: 1 to: 1

Connection probability (0-1): 1.0 Synapse type: Izhikevich Synapse (1)

Add

Figure 22. Random1 Connection Group Builder

The parameters for this plugin are as follows:

- *Description*. A brief description of the connection group.
- *Weight range 1, weight range 2*. Two different weight ranges can be included in the connection group. The parameter 'Proportion weight range 1' sets the proportion of weight range 1 that is used to create the weights. Connection weights are selected at random from the specified ranges.
- *Delay*. The delay of the connection. Delays are selected at random from the specified range.
- *Synapse type*. The type of synapse used in the connection.

7.5 Simulation Plugins

7.5.1 Nemo CUDA Simulator

NOTE: This plugin is still under development and does not work!

The Nemo CUDA Simulator plugin (see Figure 23) wraps the Nemo simulator (<http://www.doc.ic.ac.uk/~akf/nemo/index.html>).

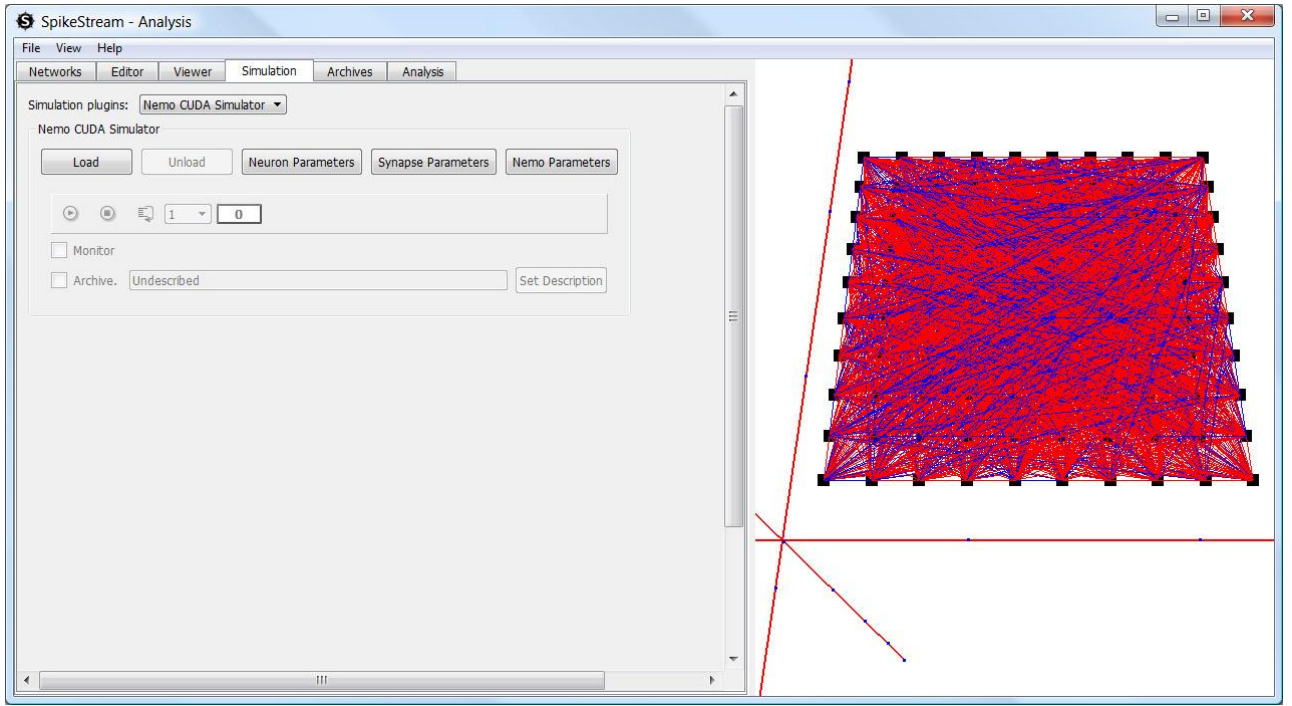


Figure 23. Wrapper for the Nemo CUDA Simulator

It has dialogs for setting the neuron and synapse parameters and the user can play or step through the simulation, monitor the results and save the firing patterns to the database. Full documentation of these features will be included with the working release.

7.6 Analysis Plugins

7.6.1 Liveliness Analyzer

The Liveliness Analyzer (see Figure 24) analyzes the currently loaded network for liveliness, which is an alternative measure of information integration. The original documentation on liveliness can be found in Aleksander (1973) and Aleksander and Atlas (1973) and some recent work in this area is available in Aleksander and Gamez (2009), Gamez and Aleksander (2009) and Aleksander and Gamez (2010).

In earlier work on liveliness, the liveliness was averaged over all states of the network, whereas the Liveliness Analyzer plugin calculates the liveliness of the network for a particular state. Liveliness in this context is the probability that a particular connection will influence the firing state of the neuron that it is connected to at the next time step. This probability is 0 or 1 in a deterministic system. The liveliness of a neuron is the sum of the liveliness of the connections to the neuron, which provides a measure of the amount of information that is integrated by the neuron at that time step.

Neurons connected by lively connections at a particular time step form a *cluster*. The liveliness of a cluster is given by Equation 1.

$$\lambda_c = \lambda_{tot} \frac{\lambda_{tot}}{n^2}, \quad (1)$$

where λ_{tot} is the sum of the livelinesses of the neurons in the cluster, and n is the number of neurons in the cluster. n^2 is the maximum possible liveliness of the cluster – a situation in which all of the neurons in the cluster are connected with maximally lively connections.

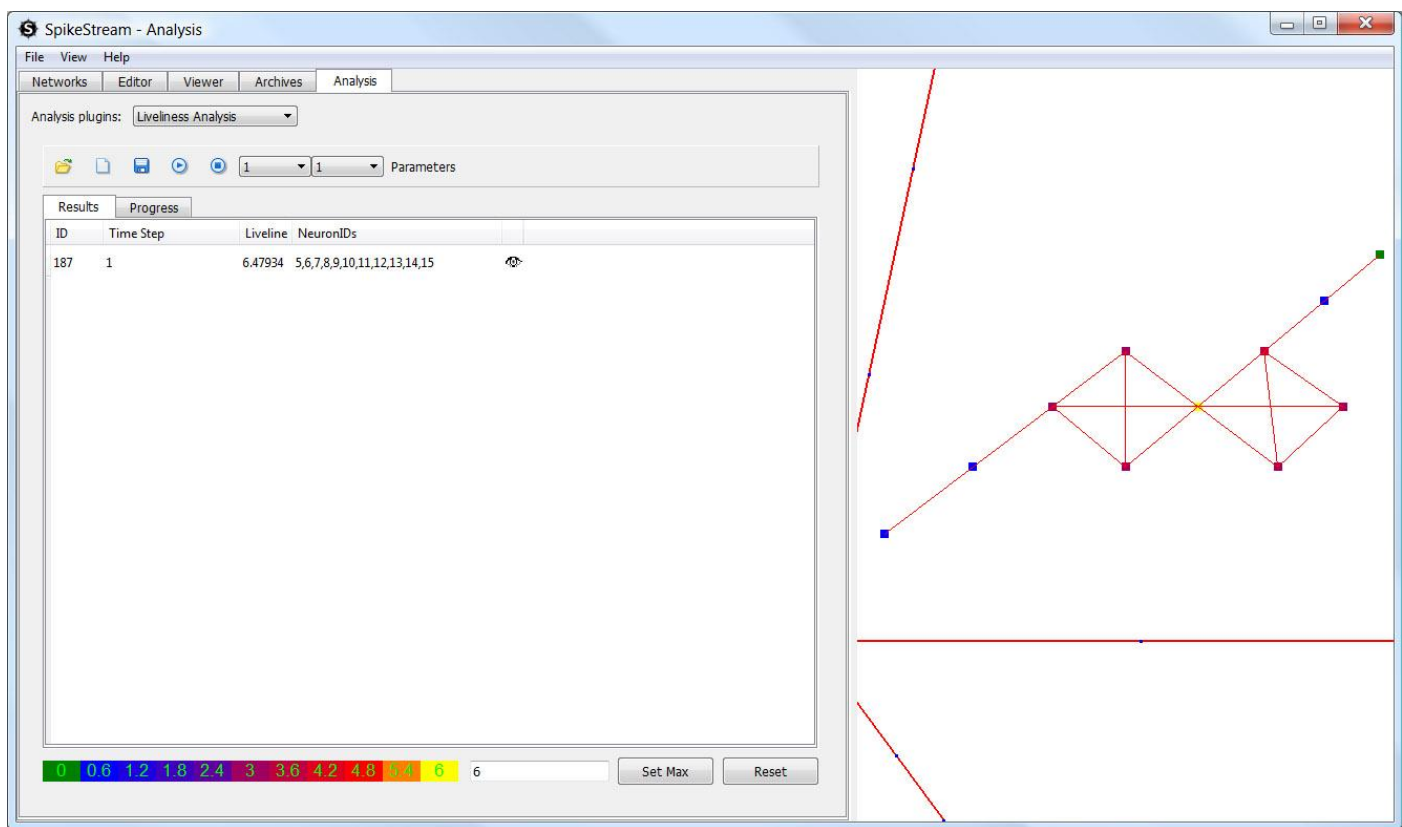







Figure 24. Liveliness Analyzer

The liveliness of each neuron is displayed as a heat map, and the scale of the heat map is displayed at the bottom of the Liveliness Analyzer plugin. This scale starts at zero and the default maximum is the maximum liveliness of any neuron in the current analysis. This maximum can be changed by entering a number next to the “Set Max” button and clicking on “Set Max”. Changing the maximum value of the scale enables comparison between heat maps from different analyses.

The Liveliness Analyzer has a toolbar with the following controls:

-  - Opens an existing analysis
-  - Creates a new analysis
-  - Starts the analysis running
-  - Stops the analysis running.
-  - Launches a dialog that enables you to save the analysis as a tab-separated text file.

The drop down combo boxes enable the selection of the time steps from the archive that are going to be analyzed. A single time step can be analyzed, or a range of time steps. These controls are only enabled when a network and an archive are loaded.

The “Parameters” button launches a dialog that enables certain parameters of the analysis to be set:

- *Analysis description.* A description of the analysis.
- *Number of simultaneous threads.* Each time step is analyzed using a separate processing thread. This parameter sets the maximum number of threads that run simultaneously.

- *Generalization*. Sets the generalization of the weightless neurons in the network. See Aleksander (2005) for more information about this parameter.
- *Store_connection_liveliness_as_temporary_weights*. As the analysis runs the liveliness of each connection is stored as the temporary weight of that connection. These weights can be viewed using the Viewer Tab. The temporary weights are stored by each thread running for each time step, so there will be an interleaving of the temporary weights unless the analysis is only run for a single time step. This parameter is only useful if you are only analyzing a single time step and it should be disabled for large analyses.

Most of these parameters are not editable after an analysis has been run and its results stored.

The liveliness analyzer has two tabs. The “Progress” tab displays the progress of each time step that is running. The Results Tab lists the clusters that have been found. Clicking the eye symbol displays the heat map of the cluster in the 3D Network Viewer, as shown on the right side of Figure 24.

7.6.2 State-based Phi Analyzer

The State-based Phi Analyzer (see Figure 25) carries out the analysis of the currently loaded network for information integration using the algorithm described in Balduzzi and Tononi (2008).

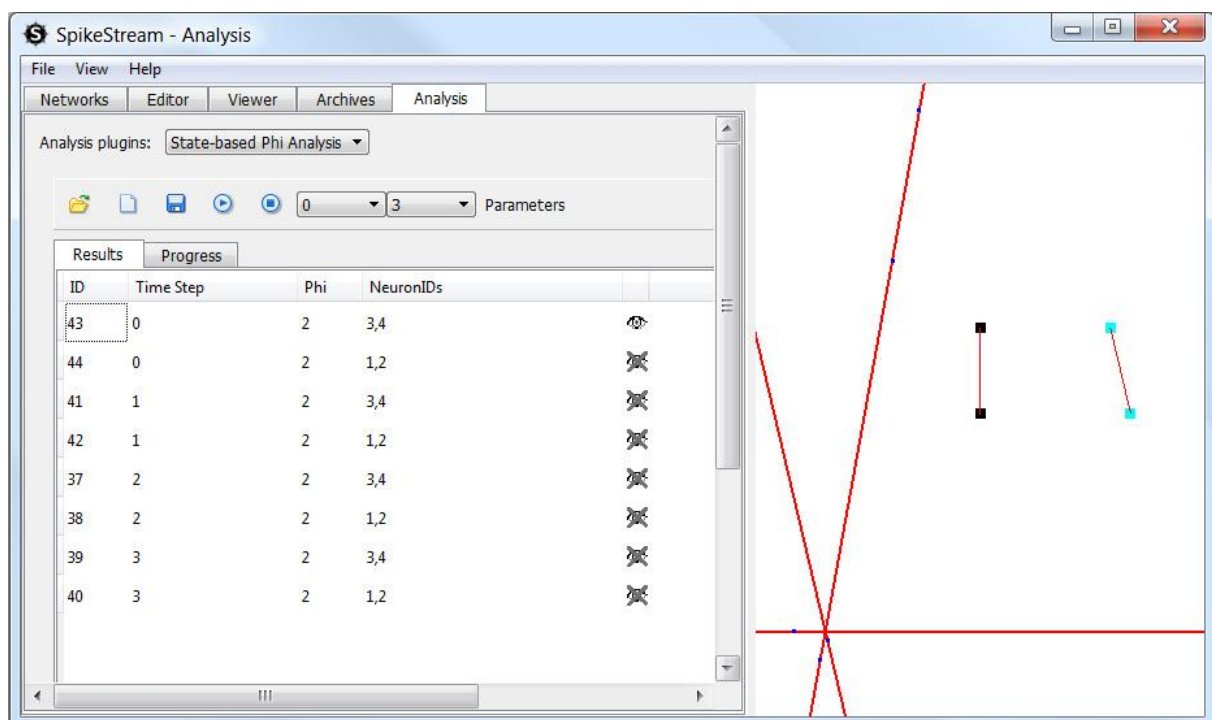


Figure 25. State-based Phi Analyzer Plugin

The State-based Phi Analyzer has a toolbar with the following controls:

- Opens an existing analysis
- Creates a new analysis
- Starts the analysis running
- Stops the analysis running.
- Launches a dialog that enables you to save the analysis as a tab-separated text file.

The drop down combo boxes enable the selection of the time steps from the archive that are going to be analyzed. A single time step can be analyzed, or a range of time steps. These controls are only enabled when a network and an archive are loaded.

The “Parameters” button launches a dialog that enables certain parameters of the analysis to be set:

- *Analysis description.* A description of the analysis.
- *Number of simultaneous threads.* Each time step is analyzed using a separate processing thread. This parameter sets the maximum number of threads that are running at a particular point in time.
- *Generalization.* Sets the generalization of the weightless neurons in the network. See Aleksander (2005) for more information about this parameter.
- *Ignore_disconnected_subsets.* A subset will have zero ϕ if it contains isolated neurons that are not connected to any other neuron in the subset. When this parameter is set to 1 these subsets are excluded at an early stage of the analysis.
- *minimum_complex_phi.* Many networks have a large number of meaningless complexes with low values of ϕ . This parameter enables the user to filter out complexes with ϕ less than the specified value. Complexes whose phi is greater than or equal to *minimum_complex_phi* will be included in the final results. The default setting is 1.0.

Many of these parameters are not editable when the analysis has been run and its results stored.

The State-based Phi Analyzer has two tabs. The “Progress” tab displays the progress of each time step that is running. The Results Tab lists the complexes that have been found. Clicking on the eye symbol displays the highlighted complex in the 3D Network Viewer, as shown on the right hand side of Figure 25.

NOTE: The Balduzzi and Tononi (2008) algorithm has factorial dependencies and takes an extremely long time to run on large networks. A graph showing the measured and estimated performance on randomly connected networks of increasing size is given in Figure 26.

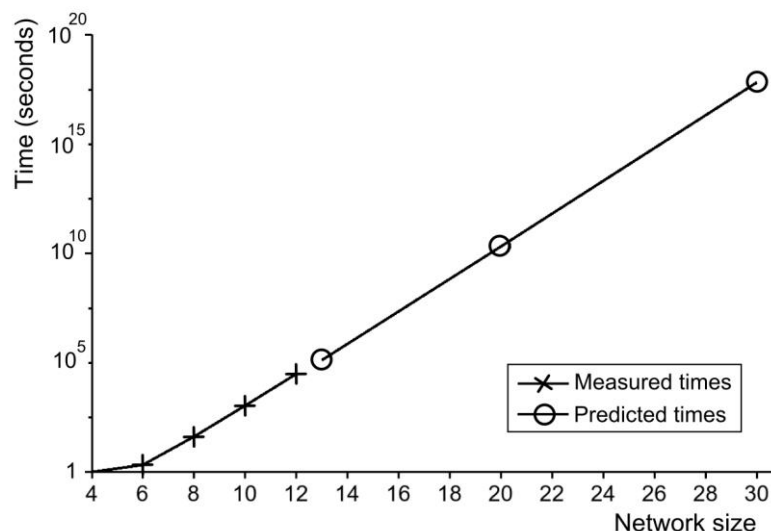


Figure. 26. Measured and predicted times for the calculation of information integration on different sizes of network using Balduzzi and Tononi’s (2008) algorithm and a Pentium IV 3.2 GHz single core computer. Each neuron in the network was randomly connected to five other neurons and their truth tables had five entries. The results are for the analysis of a single time step with a random firing pattern.

7.7 Writing SpikeStream Plugins

Plugins are written in C++ and they must extend the Qt QWidget class. Each plugin should be built as a library with the extension .dll or .so, depending on the platform. This library must be placed in the appropriate folder, depending on the type of plugin. These folders are listed in Section 7.1.

Each library must implement two external C functions, getName() and getClass(). getName() returns a unique QString describing the library; getClass() returns a class that inherits from QWidget and forms the main display widget of the plugin. For example, the State-based Phi Plugin implements these two functions as follows:

```
//Functions for dynamic library loading
extern "C" {
    /*! Creates a StateBasedPhiWidget class when library is dynamically loaded. */
    StateBasedPhiWidget* getClass() {
        return new StateBasedPhiWidget();
    }

    /*! Returns a sensible name for this widget */
    QString getName() {
        return QString("State-based Phi Analysis");
    }
}
```

spikestreamapplicationlibrary contains a number of abstract classes that can be used to develop analysis plugins. More documentation will follow soon.

The SQL for each plugin should be added to the appropriate part of the database folder. For example, SQL for the SpikeStreamNetwork database should be placed in database/network/plugins, and it should be provided in both standard and test versions. This SQL will be executed by the SpikeStream Database Configuration Tool after the database(s) have been added.

The best way of understanding how to write a plugin is to look at the code for the current plugins.

8. IMPORTING FROM NRM

8.1 Introduction

SpikeStream has a limited ability to import files from the NRM neural simulator. Networks of weightless neurons can be simulated and trained in NRM and then imported into SpikeStream for analysis. This import functionality is pretty basic and has the following limitations:

- Only files created by the most recent version of NRM (2003) can be imported into SpikeStream.
- Only random connections are supported.
- Only neural layers with a single colour plane can be imported.

8.2 Creating Files in NRM

Three files are needed for an import from NRM:

- **Configuration file (*.cfg).** Specifies the network and connectivity of the NRM network.
- **Training file (*.ntr).** Contains the training of the neurons in the network.
- **Data set file (*.set).** Contains at least one firing pattern of the network.

Configuration files can be saved in NRM by selecting **Configuration->Save As**. Once a network has been built and trained, its training can be saved by selecting **Network->Training->Save network training as**. The export of data sets is more complicated because NRM does not support the saving of the state of the network at a particular time step, and so a convention has to be used in which the state of each layer in the network is added in sequence to the data set to store the network's state at a particular point in time. For example, consider the network shown in Figure 27.

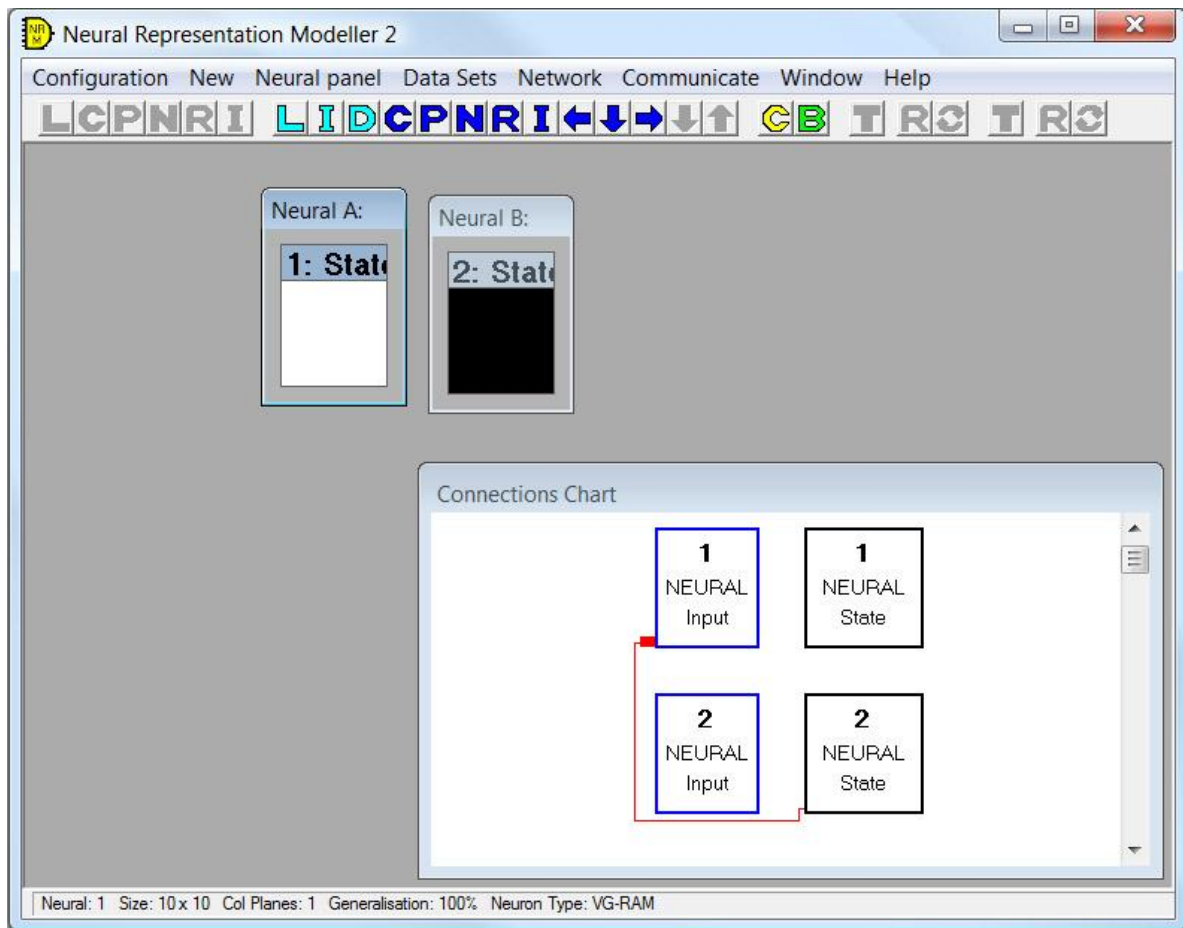


Figure 27. NRM network at time t_1

To add the state of this network, click on **Data sets->Create new**. This creates a new set and adds the state of Neural A to the set. Next, select Neural B and click on **Data sets->Add data array**. This adds the state of Neural B to the set. Suppose that at a later point in time the network is in the state shown in Figure 28.

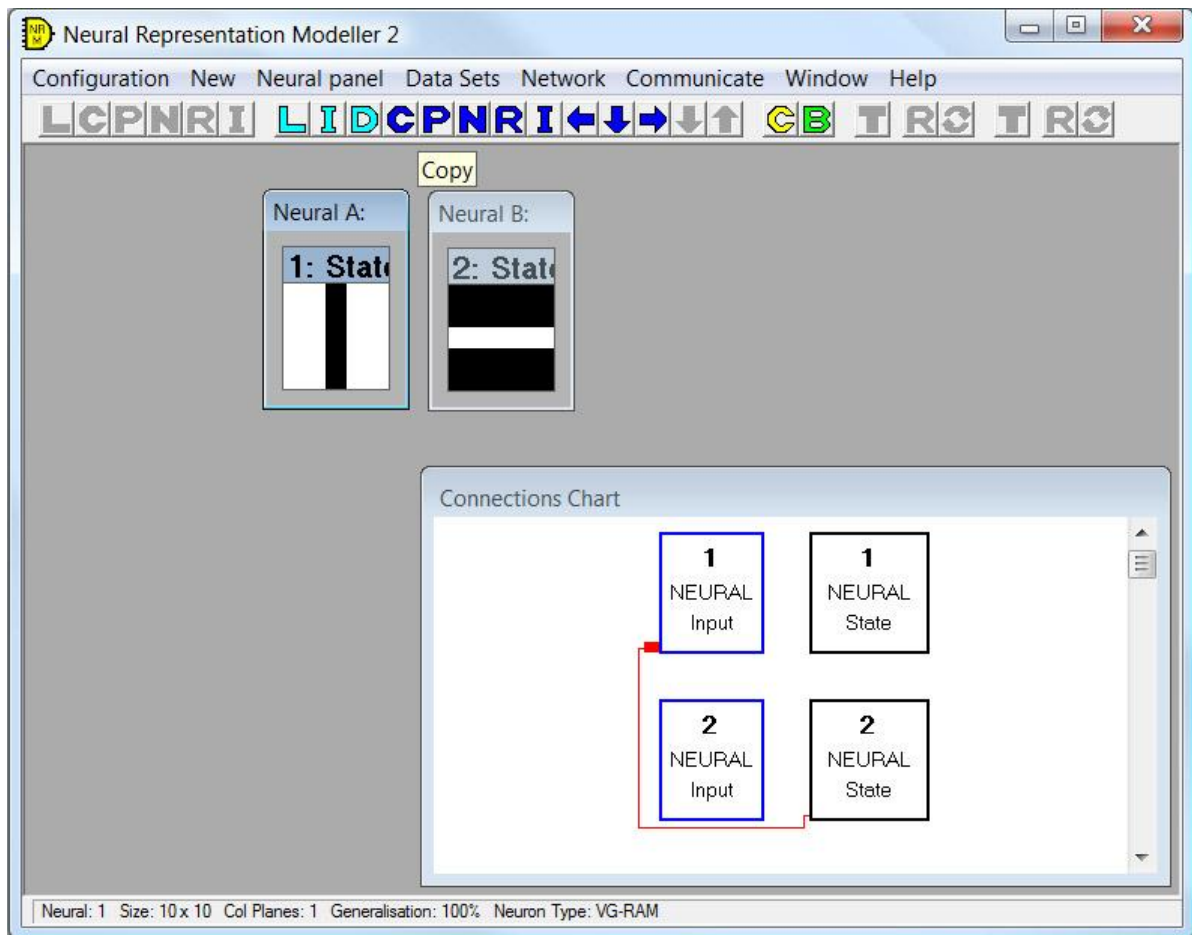


Figure 28. NRM network at time t_2

To add this state of the network to the set, select Neural A and click on **Data sets->Add data array**. Select Neural B and click on **Data sets->Add data array**. Your data set should now contain the following four entries

1. Neural A at time t_1 (all white).
2. Neural B at time t_1 (all black).
3. Neural A at time t_2 (vertical black stripe on white background).
4. Neural B at time t_2 (horizontal white stripe on black background).

Click on **Data sets->End create and save** to save this data to a .set file. When SpikeStream imports this data set it will interpret the first entry as the state of the first layer at time t_1 , the second entry as the state of the second layer at t_1 , and so on until it runs out of layers. The next entry will then be interpreted as the state of the first layer at t_2 , and so on.

8.3 Importing NRM Files into SpikeStream

Once you have created your configuration, training and data set files, you can import them into SpikeStream by clicking **File->Import NRM Network** or using the shortcut **CTRL + M**. This displays the dialog shown in Figure 29.

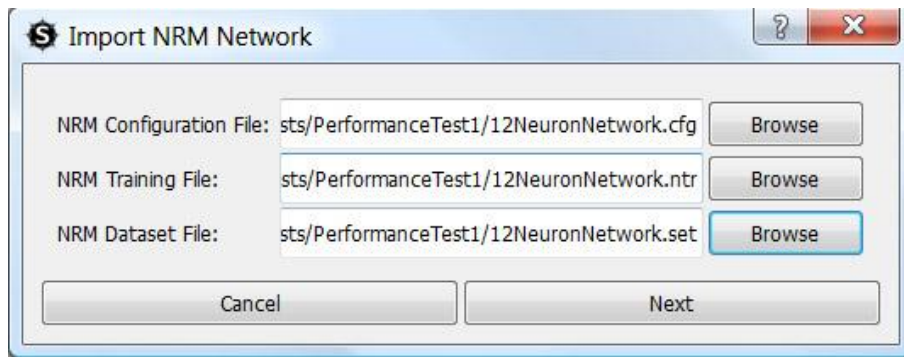


Figure 29. First screen of NRM Import Dialog.

Select the configuration, training and data set files that you want to import and click “Next”. If the files match and can be imported, you will be presented with the dialog shown in Figure 30.

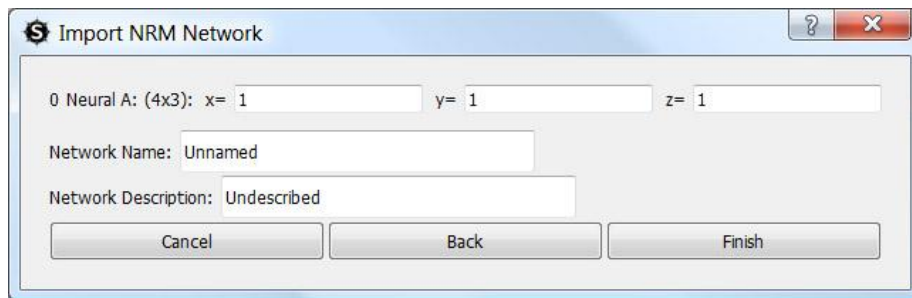


Figure 30. Second screen of NRM Import Dialog

This part of the dialog enables you to select the 3D location of the layers that are being imported and to provide a name and description of the network. When you click on “Finish”, the NRM data will imported into the SpikeStream database as a new network, which will appear in the list of networks on the Network Tab (see Section 6.1).

9. DATABASES

SpikeStream is based around three MySQL databases and cannot run without them. These databases are used to store different types of information:

- **SpikeStreamNetwork**. Holds information about the networks, neurons and connections.
- **SpikeStreamArchive**. Holds firing patterns of the network for each time step.
- **SpikeStreamAnalysis**. Stores the results of analyses of the network.

These databases can be hosted anywhere in the world as long as the firewalls are configured correctly.

If you want to run the unit tests, then you will also need a matching set of three databases: **SpikeStreamNetworkTest**, **SpikeStreamArchiveTest** and **SpikeStreamAnalysisTest**.

More information about the structure of these databases can be found by looking at their SQL, which is in the database folder of the SpikeStream distribution. The SpikeStream databases can be manually installed by running this SQL.

SpikeStream comes with a database configuration tool that makes the setting up of the databases easier (see Section 4). Before running this tool, you need to install MySQL and configure it so that you know the username and password of an account that has enough privileges to create and modify databases. Instructions on how to do this on Windows and Linux are given in Section 2.2 and Section 3.2.

The database host, username and password can be manually entered in the `spikestream.config` file, which is at the root of the SpikeStream installation (see Section 10).

10. CONFIGURATION

10.1 Introduction

SpikeStream comes with a configuration file, which holds its settings. The file is called “spikestream.config” and it is at the root of the SpikeStream installation. Ignore the spikestream.config.template file, which holds the default settings.

SpikeStream ignores comment lines that start with the hash ‘#’ character and blank lines. A configuration setting consists of a parameter, for example “spikeStreamArchiveHost” followed by an equals ‘=’ sign, followed by the value of that parameter. SpikeStream needs to be restarted for changes to take effect.

If you mess the configuration file up, you can copy the settings from the spikestream.config.template file or delete spikestream.config and run the SpikeStream Database Configuration Tool.

10.2 Database Settings

The host, username and password of each database are stored in spikestream.config and the SpikeStream Database Configuration Tool (see Section 4) writes the settings that you enter to this file. The parameters are as follows:

- **spikeStreamNetworkHost.** Host of the SpikeStreamNetwork database.
- **spikeStreamNetworkUser.** Username for the SpikeStreamNetwork database.
- **spikeStreamNetworkPassword.** Password for the SpikeStreamNetwork database.
- **spikeStreamArchiveHost.** Host of the SpikeStreamArchive database.
- **spikeStreamArchiveUser.** Username for the SpikeStreamArchive database.
- **spikeStreamArchivePassword.** Password for the SpikeStreamArchive database.
- **spikeStreamAnalysisHost.** Host of the SpikeStreamAnalysis database.
- **spikeStreamAnalysisUser.** Username for the SpikeStreamAnalysis database.
- **spikeStreamAnalysisPassword.** Password for the SpikeStreamAnalysis database.

10.3 Other Settings

Other settings available in the configuration file are:

- **default_file_location.** Default location for loading files etc.
- **vertex_size.** The size of neurons in the 3D Network Viewer. It is easier to see the colour of neurons if you increase the size. On large networks smaller vertex sizes are better.
- **draw_axes.** Shows or hides the axes.
- **maximize_gui.** Controls whether the graphical interface is launched in a maximum state.
- **number_insert_connection_buffers.** Database optimization parameter. Recommended to leave it at its default setting.

11. KEYBOARD SHORTCUTS

Network Viewer Navigation

Arrow Up: Move camera positively along the Z axis

Arrow Down: Move camera negatively along the Z axis

Arrow Left: Move camera negatively along the X axis

Arrow Right: Move camera positively along the X axis

Page Up: Move camera positively along the Y axis.

Page Down: Move camera negatively along the Y axis

CTRL + Arrow Right: Rotate camera anticlockwise around Z axis

CTRL + Arrow Left: Rotate camera clockwise around Z axis

CTRL + Arrow Up: Rotate camera clockwise around X axis

CTRL + Arrow Down: Rotate camera anticlockwise around X axis.

NOTE: In its initial state, the 3D Network Viewer is organized with positive Z axis pointing upwards, the positive Y axis pointing away from the viewer and the positive X axis moving left to right. Clockwise and anticlockwise are from the point of view of looking down the axis towards zero. The easiest way of learning how to navigate is try out the keys and observe their effect.

Other Shortcuts

CTRL + R: Resets the view to its initial position.

CTRL + M: Shows dialog for importing NRM network.

F1: Show Networks Tab

F2: Show Editor Tab

F3: Show Viewer Tab

F4: Show Simulation Tab

F5: Show Archives Tab

F6: Show Analysis Tab

REFERENCES

- Aleksander, I. (1973). Random Logic Nets: Stability and Adaptation. *International Journal of Man-Machine Studies* 5: 115-31.
- Aleksander, I. (2005). *The World in My Mind, My Mind in the World: Key Mechanisms of Consciousness in People, Animals and Machines*. Exeter: Imprint Academic.
- Aleksander, I. and Atlas, P. (1973). Cyclic Activity in Nature: Causes of Stability. *International Journal of Neuroscience* 6: 45-50.
- Aleksander, I. and Gamez, D. (2009). Iconic Training and Effective Information: Evaluating Meaning in Discrete Neural Networks. *Biologically Inspired Cognitive Architectures II. Papers from the AAAI Fall Symposium*. AAAI Technical Report FS-09-01, pp. 2-10.
- Aleksander, I. and Gamez, D. (2010). Informational Theories of Consciousness: A Review and Extension. Submitted to BICS 2010.
- Balduzzi, D. and Tononi, G. (2008). Integrated information in discrete dynamical systems: motivation and theoretical framework. *PLoS Comput. Biol.* 4(6).
- Gamez, D. and Aleksander, I. (2009). Taking a Mental Stance Towards Artificial Systems. *Biologically Inspired Cognitive Architectures II. Papers from the AAAI Fall Symposium*. AAAI Technical Report FS-09-01, pp. 56-61.