NeuroManager User and Programmer's Guide – 0.990 Addendum (Investigation Database Upgrade)

David B. Stockton

May 10, 2017

1 Introduction

This addendum refers to the NeuroManager software package [Stockton and Santamaria, 2015, 2016], its existing documentation [Stockton, 2015, 2016a,b], and its location on GitHub [Stockton and Santamaria, 2017b]. This software addition was introduced in the ABI Tools paper (no citation yet).

This addendum supplements the documentation by describing support for an "investigation database". This new addition to the NeuroManager software allows the user to create a searchable, exploitable MySQL database from the essential elements of a neuroscience simulation project, preserving the associations between input parameter vectors, simulation results, feature extractions, and experiment/simulation feature comparisons.

Note: because of the currently local nature of the investigation database, and due to time limitations, we have not included rigorous security for obtaining a connection to the investigation database. This should not be an issue for database service from a personal or unshared computer; however in other situations the user may wish to consider supplementing the database connection methods used here. We anticipate providing an improved connection approach in the near future.

2 Concepts

An *investigation* is a new concept in NeuroManager, made possible by the addition of the *investigation database*, which accumulates simulation results that are the output of NeuroManager *sessions*. The user starts an investigation merely by establishing an investigation directory and assigning it as the results directory in the NeuroManager nmDirectorySet. The user then creates an investigation database object, which provides access to the investigation database, and attaches it to NeuroManager. If the user doesn't attach an investigation database to NeuroManager, the database is not used and NeuroManager operation proceeds normally. By initializing the database during the first NeuroManager session, then not initializing it for subsequent sessions, the user will accumulate the results of each subsequent session in the investigation database. After each session, a snapshot of the database is saved in the investigation directory; before each session, the user has the option of reloading a previous version of the database, including the most recent. In summary, an investigation involves a separate directory in which are the results of possibly more than one session, and is composed of a database together with those sessions.

In addition, at any time the user can load any of the snapshots into the database and work with it using a database access tool such as MySQL Workbench [MySQL, 2017].

These new additions make it easier to utilize, protect, and recover the often time– and resource–consuming work put into running multiple simulations, many of which can take hours or days to run.

3 Design

The investigation database consists of several tables, each of which holds session information information of a focused type. By adding foreign key links between tables, we have enabled highly detailed queries about database contents. For example, the user can identify a specific comparison and extract the simulation run involved, its input parameter vector, what machine it was run on, the version number of the simulator involved, what specific feature values were extracted, the raw data from the simulation, and even a multitude of intermediate scripts and other files.

The *sessions* table holds specific information about each NeuroManager session that is part of the investigation. Each session's sessionID (which includes its date/time information), and all associated host directories are recorded in this table.

The *machines* table holds the name and resource type (e.g., SLURM cluster) for the machines used in the investigation. This table is used to identify on which machine a simulator was placed. It is ok if the machine mix changes from session to session within an investigation.

The *simulators* table holds the name, type, and version of each simulator, as well as a link to the machine on which it was hosted.

The *ipvs* table records each input parameter vector used in an investigation. Since each row has a unique key, this differentiates ipvs that have the same values but may link to different expDataSets.

The *simulationRuns* table holds the sessionID, simSetID,and simID for the simulation, thus forming a unique set of external identifiers for each simulation in an investigation. In addition, each simulationRun has its own unique key within the table, allowing for faster, easier joins and searches. Each simulationRun also holds directory paths and filenames for raw data and other results, the state of the simulation which is updated during the session, and other simulationRun–specific data. A system of four foreign keys link this table to its associated IPV, the simulator on which it was run/is running, the session in which it was run, and the features set extracted from its data. In a sense this is the central table of the database.

The *simFeatureExtractions* table holds the features extracted from the raw data output of a simulation. Since the features are extracted remotely, this approach allows easy access to the features on the host with very little computational load.

The *expDataSets* table is for use with experimental data. It holds identifying information about the specimen/data involved, the sampling rate involved, and has room for experiment characteristics that the user has selected for use in associated simulations. For example, if an experiment feature "tau" will be used in simulations after some normalization, the original tau from the experiment can be placed in this table, while the normalized tau becomes part of the input parameter vector of the simulation.

4 Classes

Investigation database operations and tables are provided using an object–inheritance approach, so that the user can modify or override the base class implementations as needed. We have provided a minimal **investigationDB** class which offers basic functionality; each subclass will provide application–specific details as demonstrated in the MMInvDB example discussed below. In general, there are invariant tables/methods (sessions, machines, simulators) and easily overridden tables/methods (ipvs, simulationRuns). NeuroManager itself uses the basic interface; we have been careful to avoid application–specific calls within the NeuroManager core. Subclasses provide specific details as well as additional methods that the user wishes to employ external to NeuroManager.

As an example of user specification we have provided the **abiCompDB** subclass which overrides table design, adds new tables, and adds/overrides specific methods to allow addition of simulation feature extractions, experimental feature extractions, and comparisons to the database (expDataSets, simFeatureExtractions, and comparisons).

NeuroManager itself holds the handle to the investigation database, which must be constructed externally as shown in the example script discussed in Section 6.

5 Peripheral classes and files

To provide basic fully automated investigation involving an external database of experimental results and the investigation database, we developed some tools for accessing the Allen Brain Institute Cell Types database [Stockton and Santamaria, 2017a] and also constructed a few rudimentary classes that we are including with the investigation database code. The *metaheuristic* classes provide (for now) a simple grid search, and the *comparator* classes provide customizable comparison between experimental and simulation extracted features. These classes can be found on the GitHub NeuroManager site [Stockton and Santamaria, 2017b].

5.1 Comparator

The **comparator** class is an abstract base class that uses the ABI specimen, experiment, and experiment info terminology; however this should be essentially universal so we have not tried to pull out this minor bit of ABI–ness. The **comparator** class holds information for the two databases involved (simulation and experimental). Subclasses of **comparator** such as **Cmp01** provide specific comparisons. To provide efficiency, each **comparator** subclass can do up to five different comparisons using the features that it pulls from the databases; each comparison is placed into the investigation database as a column in a comparison row along with the identity of the comparison subclass. This approach allows significantly different comparisons (such as single versus multiple versus sweep sets) to be captured as subclasses while allowing feature–set differences to be captured within a single subclass. **Comparator** etc are found in the InvDBUtils directory under the NeuroManager installation directory.

5.2 Metaheuristic

The **metaheuristic** class is a rudimentary abstract base class that provides a first–draft interface for metaheuristic algorithms to interact with NeuroManager–based scripts. The basic approach is to construct the class using subclass–specific initialization data, then use getPointSet() each generation to get the set of points to be simulated. We have not yet added a way for objective function results to feed back into the algorithm (hence "rudimentary") and our only subclass at present is the **explicitGrid** subclass, which provides the Cartesian product of all cell arrays provided in the initialization data. The points are provided all at once in the first generation.

5.3 SimSpecFile

The **SimSpecFile** class, held in the Core directory, is a subclassable base class that builds a SimSpec file for the user in a compact form that simplifies scripts. It is designed for automatic use in conjunction with the **metaheuristic** class.

6 MMInvDB: An example investigation script

We have constructed a working example called "MMInvDB" which is an extension of an original NeuroManager example called "KhStudy" — see the User's Guide [Stockton, 2015]. In that example, we adapted a model published on ModelDB for use with NeuroManager, then constructed a script and Simulator for NeuroManager that adds to the input parameter vector the stimulus current, stimulus duration, and the conductivity of the Kh channel in soma, smooth dendrite, and spiny dendrite.

For this example, we added to KhStudy the ability to modify the conductivity of the CaE and KD channels in those same three locations, as well as remote feature extraction from the resulting raw data files using the ABI SDK feature extraction classes (see Stockton and Santamaria [2017a]). We then extended the script with ABI database access, automated parameter search, the use of an investigation database, and automated comparisons between the features extracted from the ABI experimental data and the same features extracted from the Miyasho models.

So the example accesses a local database of experimental features, runs a grid of Miyasho model simulations against the experimental data, performs comparisons between them, finds the best matches, and places all the resulting findings into the experimental database. Figure 1 shows a flowchart of the example script.

7 Data type conversions

MySQL has a float limitation of 14 digits, which is not a limitation because we are only storing extracted features and pointer strings in the database. The actual simulation raw data is processed on the remote using MATLAB and python/NumPy, then downloaded to the host. In addition, the simulation raw data files can be accessed using the path and filename information stored in the MySQL database as part of the simulation table.

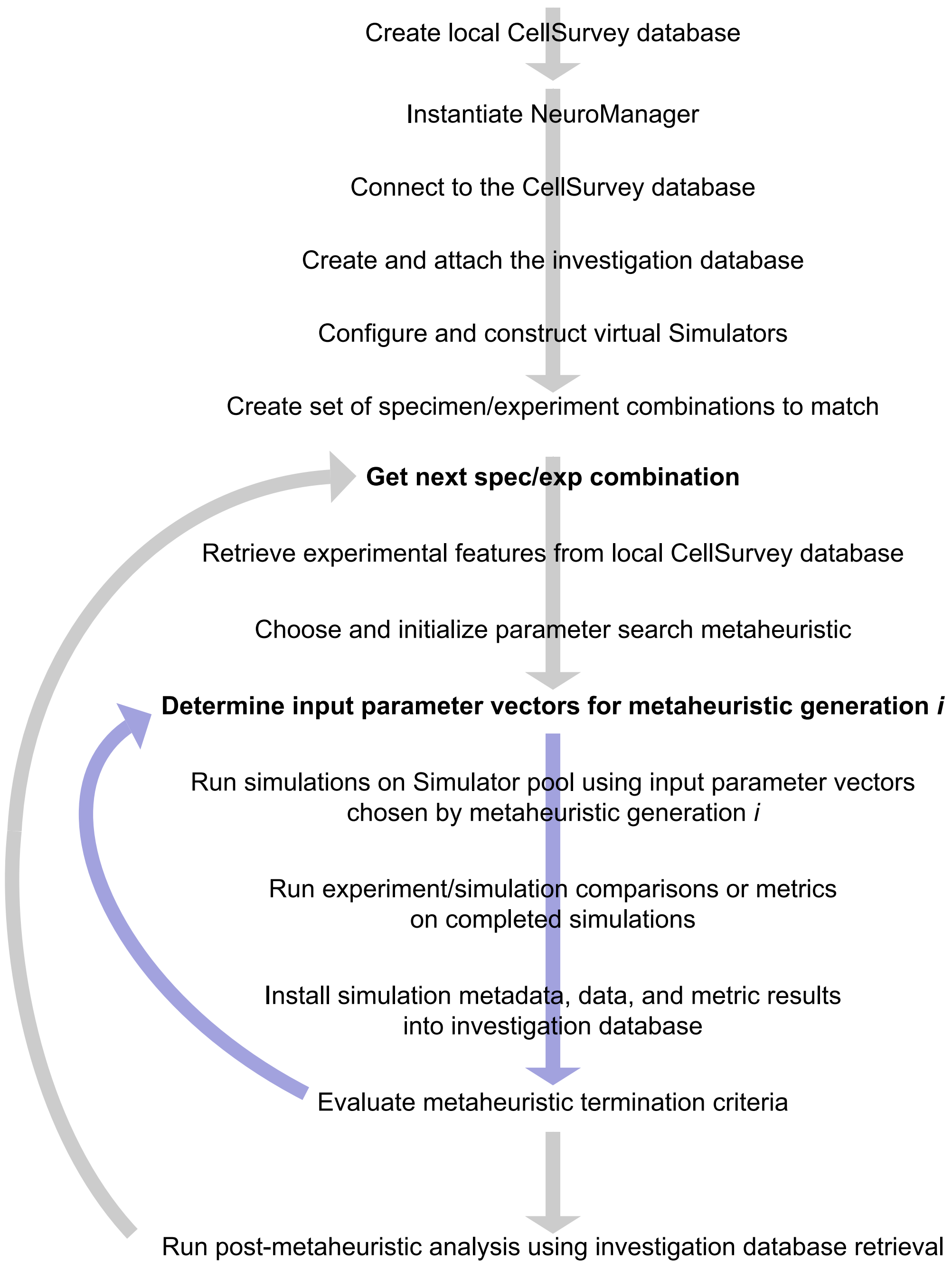
Create local CellSurvey database
Instantiate NeuroManager
Connect to the CellSurvey database
Create and attach the investigation database
Configure and construct virtual Simulators
Create set of specimen/experiment combinations to match
Get next spec/exp combination
Retrieve experimental features from local CellSurvey database
Choose and initialize parameter search metaheuristic
Determine input parameter vectors for metaheuristic generation i
Run simulations on Simulator pool using input parameter vectors chosen by metaheuristic generation i
Run experiment/simulation comparisons or metrics on completed simulations
Install simulation metadata, data, and metric results into investigation database
Evaluate metaheuristic termination criteria
Run post-metaheuristic analysis using investigation database retrieval

Figure 1: Flowchart of example "MMInvDB". The first line uses the Python CellSurvey tool to create a local database of experimental features from the ABI Cell Types Database. A series of MATLAB statements then run simulations against experimental data from that local database according to a search metaheuristic and perform comparisons on the features extracted from the simulations. The script is a basic example of a fully automated neuroscience investigation that is fully documented with logs and a series of database snapshots.

References

MySQL. MySQL Workbench, 2017. URL https://www.mysql.com/products/workbench/.

David Stockton and Fidel Santamaria. ABI Tools Website, 2017a. URL http://github/SantamariaLab/ABITools.

David Stockton and Fidel Santamaria. NeuroManager Website, 2017b. URL https://github.com/SantamariaLab/NeuroManager.

David B. Stockton. NeuroManager 0.961 User and Programmer's Guide, 2015. URL https://github.com/SantamariaLab/NeuroManager/blob/master/UserGuide/UserGuideRev6PostReview.pdf.

David B. Stockton. NeuroManager User and Programmer's Guide – 0.980 Addendum, 2016a. URL https://github.com/SantamariaLab/NeuroManager/blob/master/UserGuide/NMUGAddendum.pdf.

David B. Stockton. NeuroManager User and Programmer's Guide – 0.985 Addendum ("One Compile" Upgrade), 2016b. URL https://github.com/SantamariaLab/NeuroManager/blob/master/UserGuide/UGOneCompileAddendum.pdf.

David B. Stockton and Fidel Santamaria. Automating NEURON simulation deployment in cloud resources. *Neuroinformatics*, Sep 2016. ISSN 1559-0089. doi: 10.1007/s12021-016-9315-8. URL http://dx.doi.org/10.1007/s12021-016-9315-8.

David Bruce Stockton and Fidel Santamaria. NeuroManager: A workflow analysis based simulation management engine for computational neuroscience. *Frontiers in Neuroinformatics*, 9(24), 2015. ISSN 1662-5196. doi: 10.3389/fninf.2015.00024. URL http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2015.00024/abstract.