2014

ODM Tools Python: Open Source Software For Managing Continuous Sensor Data

Jeffery S. Horsburgh

Stephanie Reeder

Amber Spackman Jones

ODM TOOLS PYTHON: OPEN SOURCE SOFTWARE FOR MANAGING CONTINUOUS SENSOR DATA

JEFFERY S. HORSBURGH (1), STEPHANIE REEDER (2), AMBER SPACKMAN JONES (2)
(1): Department of Civil and Environmental Engineering and Utah Water Research Laboratory, Utah State University, 8200 Old Main Hill, Logan, UTAH 84322-8200, USA
(2): Utah Water Research Laboratory, Utah State University, 8200 Old Main Hill, Logan, UTAH 84322-8200, USA

Hydrologic and water quality data is being collected at high frequencies, for extended durations, and with spatial distributions that require infrastructure for data storage and management. The Observations Data Model (ODM), which is part of the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS), was developed as a framework in which to organize, store, and describe point observations data. In this paper we describe ODM Tools Python, which is an open source software application that allows ODM users to query and export, visualize, and edit data stored in an ODM database. Previous versions of ODM Tools included functionality to export data series and associated metadata, plot and summarize single data series, generate derivative data series, and edit data series using a set of simple tools. We have developed a new version of ODM Tools in Python that adds a modernized graphical user interface, multiple platform support (Windows, Linux, and Mac), multiple database support (Microsoft SQL Server and MySQL), and support for automated scripting of quality control edits performed on data series through an integrated Python script editor and console. Scripting records the corrections and adjustments made to data series in the quality control process, ensuring that the steps are traceable and reproducible. Additional improvements to ODM Tools Python include customizable queries for data selection and export, the ability to plot multiple data series simultaneously with various plot types, and user-defined functions for data series editing and derivation.

INTRODUCTION

Hydrologic monitoring with *in situ* environmental sensors presents many challenges for data management, particularly for large-scale networks consisting of multiple sites, sensors, and personnel. The high frequency, extended duration, and spatial distribution of data collection efforts require cyberinfrastructure to support and facilitate research using sensor data streams. Researchers and practitioners need tools for data import and storage as well as data access and management. In addition to addressing the challenges presented by the sheer quantity of data, monitoring network managers need practices to ensure high data quality, including procedures and software tools for data post processing and quality control.

In this paper we describe the architecture and functionality of an open source software tool called ODM Tools Python, which enables users to query and export, visualize, and edit time series observations stored in an Observations Data Model (ODM) database [1]. ODM was developed as a standard data model in which to organize, store, and describe point observations (e.g., observations made at fixed point monitoring sites such as streamflow, water quality, and weather monitoring stations) and sufficient metadata for observations to be unambiguously interpreted by multiple users. ODM is implemented in relational database software to permit flexibility in querying and data retrieval. ODM Tools Python is a modernized and advanced version of the original ODM Tools software, which, along with ODM, was developed as part of the HydroServer software stack [2] and the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS) [3, 4].

Previous versions of ODM Tools were developed in Microsoft Visual Studio .NET, which limited deployment to Microsoft Windows-based computers. It included functionality to export data series and associated metadata, plot and summarize single data series, generate derivative data series, and edit data series using a set of simple tools. The new, Python-based version of ODM Tools adds a modernized graphical user interface (GUI) with dockable components, multiple platform support (Windows, Linux, and Mac), support for multiple relational database management systems (RDBMS) (Microsoft SQL Server and MySQL), enhanced plotting and visualization, and automated scripting of quality control edits performed on data series through an integrated Python script editor and console. Additional improvements include customizable queries for data selection and export, the ability to plot multiple data series simultaneously with various plot types, and user-defined functions for data series editing and derivation. We anticipate that the functionality that we have developed in ODM Tools Python will be widely applicable to managers of streaming sensor data and that the architecture of the ODM Tools Python software will be informative for researchers developing similar tools.

ODM TOOLS PYTHON ARCHITECTURE

The ODM Tools Python architecture consists of three layers (Figure 1). The user interface layer provides a GUI within which users can visualize and export data, generate summary statistics, and perform data quality control editing. The GUI was designed and implemented using wxPython (http://www.wxpython.org/), with matplotlib for data visualization (http://matplotlib.org/). The user interface layer also includes an integrated Python script editor and console using the PyCrust component of wxPython.

The ODM Tools Python service layer consists of a set of services containing the core functionality of the application. A service is an object that creates and receives data objects defined in the data access layer. The GUI interacts with the data layer through the service layer. For example, a Plot Generator service provides functions for creating the visualizations within the GUI, the Edit Functions service provides functions for performing data quality control editing on time series data, and the Script Recorder service manages the automated recording of Python function calls during data editing sessions. We designed the Edit Functions service generally so it could be used outside of ODM Tools Python as a general scripting tool.

Finally, ODM Tools Python uses a SQLAlchemy-based data access layer (http://www.sqlalchemy.org/). This layer serves to abstract from the ODM database and provides a set of programmable objects that facilitate data management within the application – rather than programming SQL queries directly against the ODM database. This is a significant

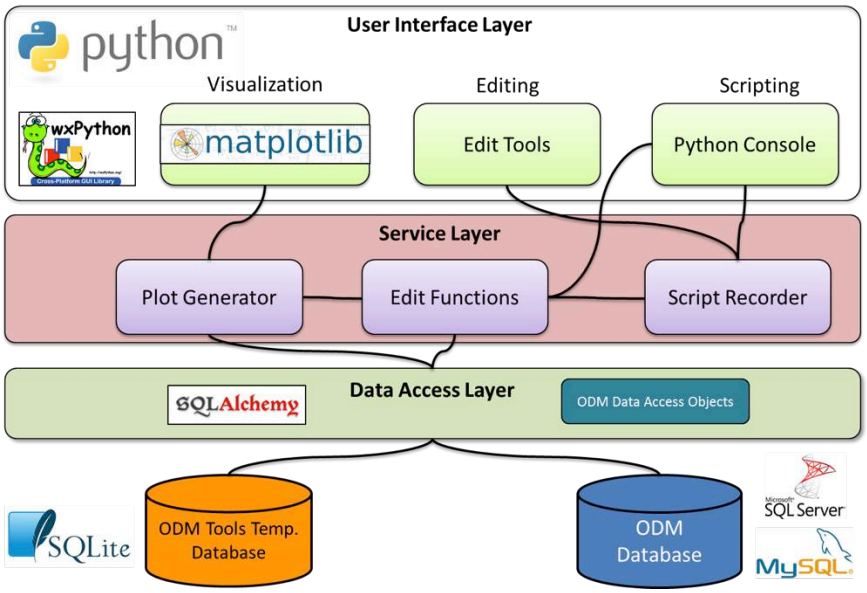

Figure 1. ODM Tools Python architecture.

improvement over earlier versions of ODM Tools, which used SQL queries spread throughout the code that were difficult to maintain. As a new feature, the data layer also provides support for connections to ODM databases within multiple RDBMS (currently Microsoft SQL Server and MySQL), whereas older versions worked only with Microsoft SQL Server. Additionally, ODM Tools Python uses SQLite as an in-memory database to temporarily store and manipulate time series data objects during data editing sessions. Additional Python modules included within ODM Tools Python code are detailed on the ODMTools Python GitHUB page (https://github.com/UCHIC/ODMToolsPython).

ODM TOOLS PYTHON GRAPHICAL USER INTERFACE

The ODM Tools Python GUI consists of a ribbon control, a data visualization panel, a time series selection panel, a table viewer panel, a Python script editor, and an integrated Python console. Figure 2 shows the standard configuration of these components upon opening the software (not all components are shown). Components shown in the window can be selected from the View menu at the top of the window. Several components of the GUI (e.g., the time series selection panel, table view panel, Python script editor, and Python console) are dockable, which enables users to arrange and configure their window according to their preference.

The time series selection panel at the bottom of the window enables users to select which data series are shown in the data visualizations or which series are selected for editing. Simple filters can be applied to the list of data series stored within the attached database to identify a subset of data series that meet particular criteria. This is especially useful where the underlying ODM database contains many data series. Users can right click on a data series listing in the time series selection panel and choose an option in the context menu to export the data or the metadata, providing a simple method for exporting data from the underlying database to a file on the user's computer.

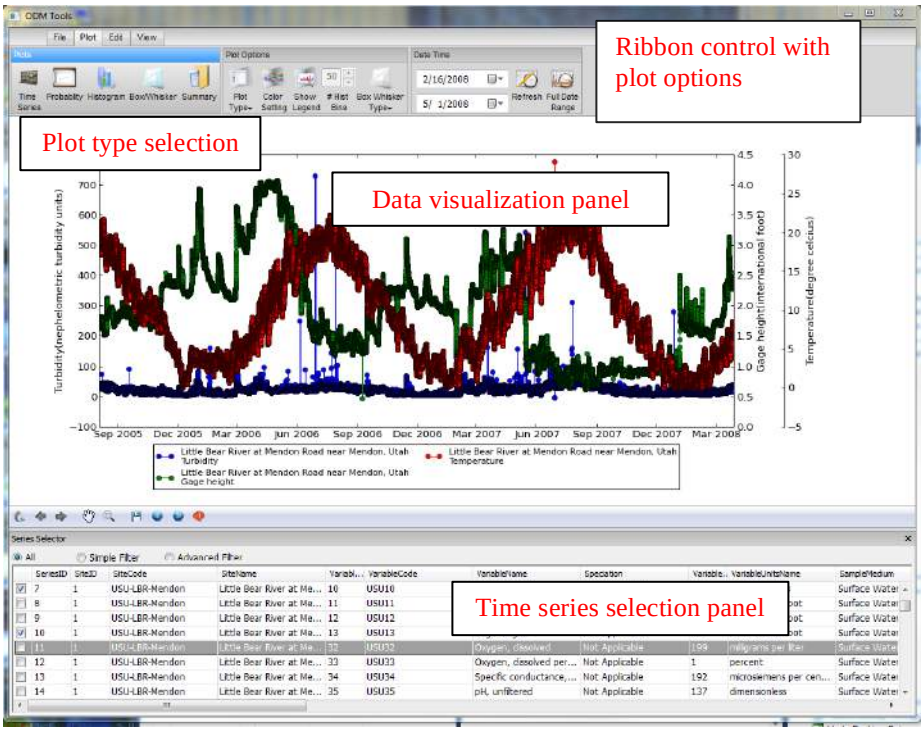

Figure 2. ODM Tools graphical user interface.

Enhanced visualizations within ODM Tools Python include the ability to plot multiple time series simultaneously (Figure 2). This is a significant improvement over previous versions of ODM Tools and is particularly useful in data quality control editing. Anomalous values in the time series being edited can easily be compared to other time series at the same site or a time series for the same variable from another nearby site to determine whether the anomalous values should be removed, interpolated, left alone, etc. Additional plot types supported by ODM Tools Python are shown in Figure 3.

DYNAMIC DATA QUALITY CONTROL EDITING

A major goal in the development of ODM Tools Python was to add the capability to track all changes made to a time series during the quality control process. Figure 4 shows common types of anomalies in raw sensor data streams and illustrates the types of corrections that are needed in the quality control process. ODM Tools Python provides several data editing buttons on the main ribbon toolbar, including value filtering, insertion or deletion of values, interpolation, linear drift correction, value adjustment, and value flagging. The new ODM Tools Python scripting interface automatically records the user's actions as they click these buttons to perform corrections and adjustments on data series in the quality control process.

Each button on the data editing toolbar is mapped to a data editing function in the ODM Tools Python service layer. When a button is clicked, it executes the underlying data editing function and fires an equivalent line of code to the Python script editor (Figure 5). This records the sequence of edits in a Python script that can be saved as a file, ensuring that the editing steps are traceable and reproducible. Table 1 lists the data editing functions available in the ODM Tools Python service layer.

Table 1. ODM Tools Python data editing functions in the data editing service.

Data Editing Function	**Description**
add new points	Inserts new data values into the data series
change data values in filter	Modifies selected data values by adding, subtracting, or multiplying by a constant value
create method	Creates a new method in the underlying ODM database
create qualifier	Creates a new data value qualifier in the underlying ODM database
create quality control level	Creates a new data processing/quality control level in the underlying ODM database
create variable	Creates a new variable in the underlying ODM database
delete filtered points	Deletes selected data values from the data series
drift correction	Applies a linear drift correction on the selected data values
export series data	Exports a selected data series to a file
export series metadata	Exports metadata for a selected data series to a file
filter by data value	Selects data values falling within a specific data value range
filter by date	Selects data values falling within a specific date range
find data_gaps	Selects data values on either side of data gaps of a given time duration
flag	Adds a data qualifying comment to selected values in the data series
interpolate	Interpolates the selected data values in the data series using the previous and next data values
reset the current filters	Removes all applied filters
restore original values	Discards any changes made and replaces the in-memory database with a new copy of the data series
return the points in the filter	Gets the data values that meet the criteria of the currently applied filter
return the points in the series	Gets the data values in the currently selected data series
toggle "filter from previous" on and off	Alerts ODM Tools to select data values from the full set or the existing selected set of data values
toggle recording on and off	Alerts ODM Tools to begin or end recording the user's data editing steps to the Python script editor
value change threshold	Selects data values where the change from one value to the next is greater than a threshold

Within ODM Tools Python, all edits are made on copies of the raw data in order to preserve the original data. When a data series is selected for editing, a copy of the data series is created within an in-memory SQLite database. All subsequent edits and modifications are made to the memory copy of the time series. When editing is finished, the time series stored in memory is written to the underlying ODM database as a new version and the memory copy is destroyed. Using SQLite as the in-memory storage for data series being edited provides a robust and convenient structure that supports querying and filtering in a memory object that does not have to be persisted on disk.

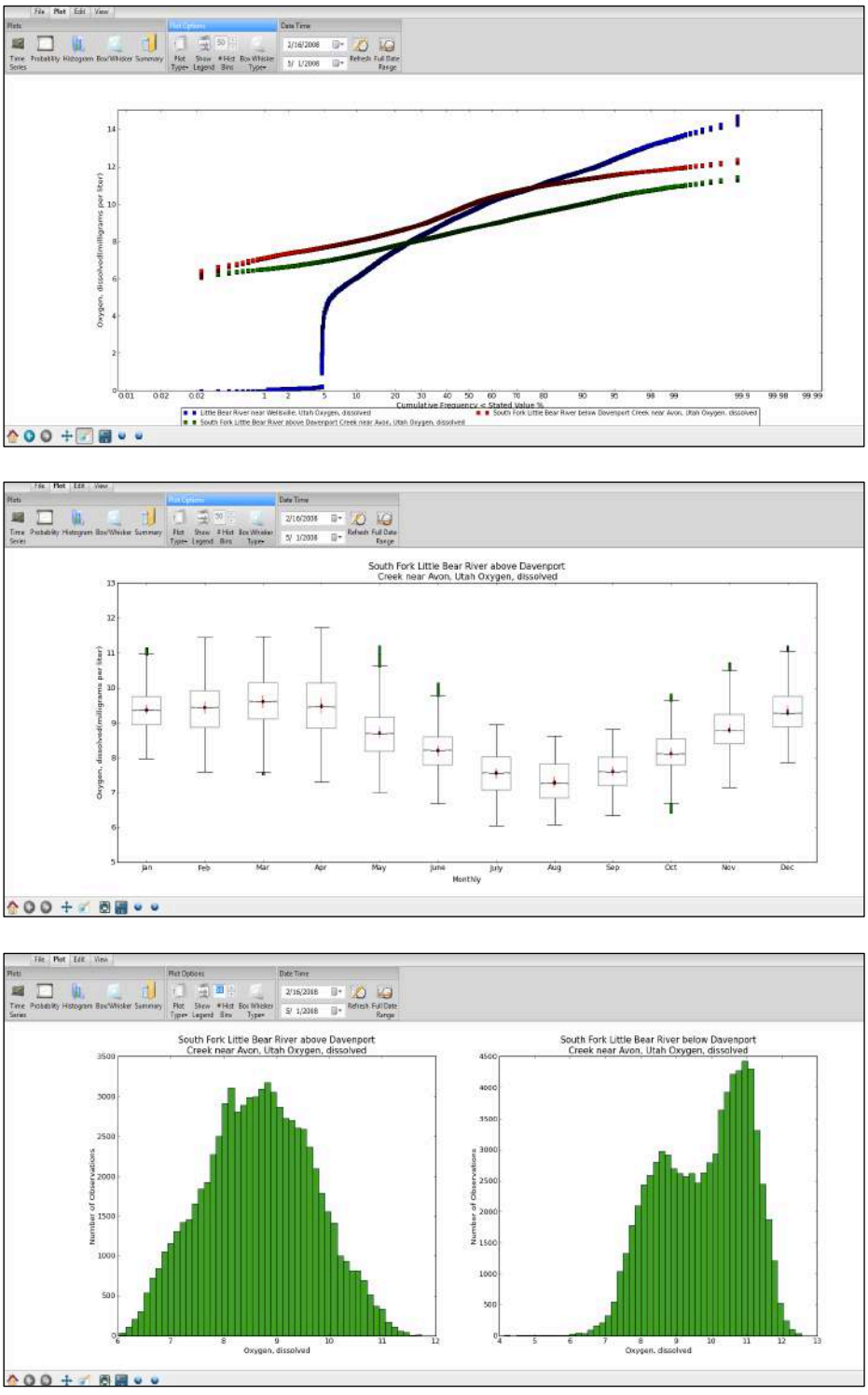

Figure 3. ODM Tools Python data visualization plot types. In addition to multiple time series plots shown in Figure 2, ODM Tools Python provides probability, box/whisker, and histogram plots. Enhanced plots in ODM Tools Python can now handle multiple time series simultaneously.

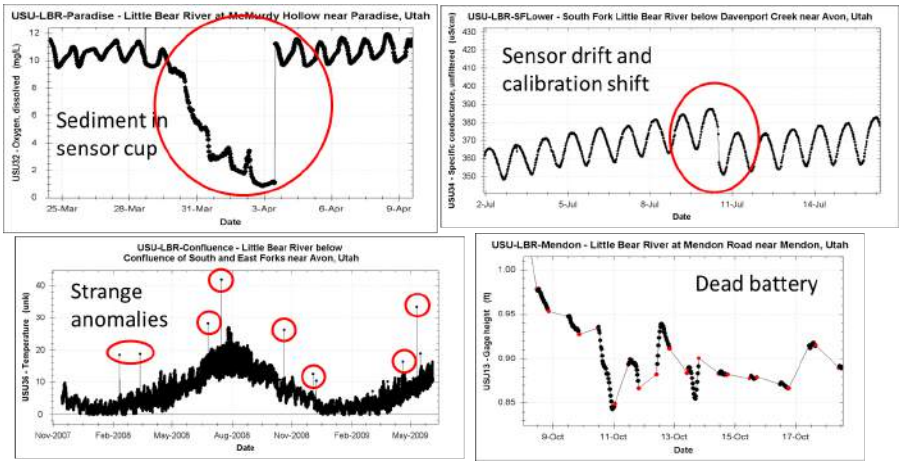

Figure 4. Example anomalous sensor data.

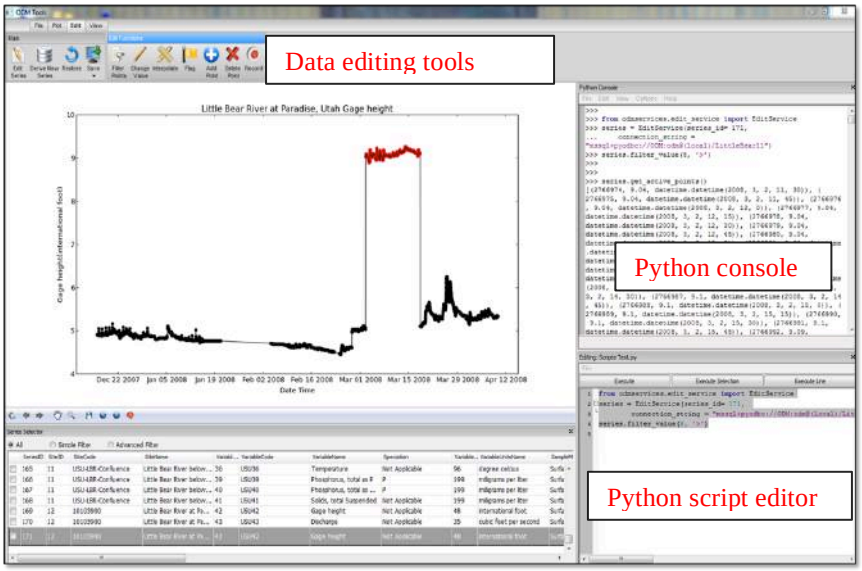

Figure 5. ODM Tools Python GUI with integrated Python script editor and console.

DISCUSSION AND CONCLUSIONS

We have developed a new version of ODM Tools in Python that provides multiple platform support, multiple database support, and support for automated scripting of quality control edits performed on data series – each of which were major limitations in the original versions of ODM Tools. Another new development is a modernization of the ODM Tools GUI. The integrated ribbon control provides more flexibility in implementing toolbar buttons and menus and is more consistent with common software with which users will be familiar. As user layout preferences may differ when visualizing data versus performing quality control editing, the dockable window components enable users to configure their window to suit their needs.

For data quality control editing, ODM Tools combines buttons exposed via the main GUI toolbar with automatically generated Python code. The strength of this approach is that it

exposes the ability to perform data editing actions through the GUI, but translates any actions within the GUI into executable Python code. The resulting script serves as a record of the edits and can be re-executed at any time to generate the same result. This approach supports novice users, who would focus on the GUI tools, and more experienced users who might go directly to scripting using the Python script editor and console.

Additional improvements to ODM Tools Python include customizable queries for data selection and export, the ability to plot multiple data series simultaneously with various plot types, and user-defined functions for data series editing and derivation. We anticipate that ODM Tools Python will be useful for data managers needing procedures and software tools for data post processing and quality control.

SOFTWARE AVAILABILITY

The software programs described in this paper are available in open source code repositories. The CUAHSI HIS HydroServer software stack, including ODM and the original versions of ODM Tools are available via the HydroServer Codeplex website and code repository (http://hydroserver.codeplex.com). The ODM Tools Python application is available on GitHUB (https://github.com/UCHIC/ODMToolsPython).

ACKNOWLEDGMENTS

This work was funded by National Science Foundation grant EPS 1208732. Additional support was previously provided by National Science Foundation Grant EAR 0622374. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Horsburgh, J.S., Tarboton, D.G., Maidment, D.R., and Zaslavsky, I., "A relational model for environmental and water resources data", *Water Resources Research*, Vol. 44, (2008), W05406, http://dx.doi.org/10.1029/2007WR006392.

[2] Horsburgh, J.S., Tarboton, D.G., Schreuders, K.A.T, Maidment, D.R., Zaslavsky, I., and Valentine, D., "HydroServer: A platform for publishing space-time hydrologic datasets". *Proc. AWRA Spring Specialty Conference on GIS and Water Resources*, Orlando, FL, (2010).

[3] Horsburgh, J.S., Tarboton, D.G., Piasecki, M., Maidment, D.R., Zaslavsky, I., Valentine, D., and Whitenack, T., "An integrated system for publishing environmental observations data", *Environmental Modeling and Software*, Vol. 24, (2009), pp 879-888, http://dx.doi.org/10.1016/j.envsoft.2009.01.002.

[4] Tarboton, D.G., Horsburgh, J.S., Maidment, D.R., Whiteaker, T., Zaslavsky, I., Piasecki, M., Goodall, J., Valentine, D., and Whitenack, T., "Development of a community Hydrologic Information System. *In: Anderssen, R. S., R. D. Braddock, and L.T.H. Newham (eds.) 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation, Modelling and Simulation Society of Australia and New Zealand and International Association for Mathematics and Computers in Simulation*, (2009), pp. 988-994, ISBN: 978-0-9758400-7-8.