SARDM: Sensitivity Analysis of Range Dynamics Models v0.6

The SARDM tool facilitates Sensitivity Analysis of Range Dynamics Model implemented using RAMAS Metapop. The tool performs the following functions:

MP Sampling Generation
The tool loads a baseline RAMAS Metapop (MP) input file and generates modified MP files for sensitivity analysis by sampling selected model parameters using either:

1. Latin Hypercube sampling (LHS) from:
    a. Uniform distribution
    b. Gaussian distribution
    c. Triangular distribution
    d. Lognormal distribution
    e. Beta distribution
2. Random sampling
3. Full factorial (lower, mid and upper values)

The tool also generates a batch file for running RAMAS Metapop with the generated MP files.

MP Results Summary
This tool loads the results of the RAMAS Metapop runs corresponding to MP files generated via MP Sampling Generation, and generates files summarising key results including some general linear model (GLM) regression analysis. The original inputs (modified MP parameters) and outputs (corresponding results) are also provided in both text and comma-separated values (CSV) formats to allow the user to conduct their own further analysis if desired. The tool also generates series plots and text and CSV data for the mean predicted Abundance and Metapopulation Occupancy (± 1 and ± 2 S.D., as well as indicating minimum and maximum values).

1 Installation Guide

The tool can be installed as a stand-alone application or utilised as a Python program when Python is installed on the user's PC.

1.1 Stand-alone application

To install the stand-alone tool on a Windows PC:

1. Run the executable installation file: *SARDM_v0_5_Windows_setup.exe*. The installation will place the program in an appropriate location on your PC (eg. *C:\Program Files*), and create a program group in the start menu, and optionally a desktop icon.
2. The installation will also install the Microsoft Visual C++ 2008 Redistributable Package, which will prompt the user to accept its terms and conditions.
3. To uninstall the tool use the Control Panel: Add or Remove Programs utility.

1.2 Running as a Python program

Assuming the user has some experience running Python programs and has Python installed on their PC, the tool can be installed as a Python program:

1. Install the following extensions for Python:
   (a) NumPy (http://www.numpy.org/)
   (b) SciPy (http://www.scipy.org)
   (c) Statsmodels (http://statsmodels.sourceforge.net/)
   (d) Pandas (http://pypi.python.org/pypi/pandas)
   (e) Setuptools (http://www.lfd.uci.edu/~gohlke/pythonlibs/#setuptools)
   (f) Patsy (http://www.lfd.uci.edu/~gohlke/pythonlibs/#patsy)
   (g) Matplotlib (http://matplotlib.org)
   (h) Pyparsing (http://pyparsing.wikispaces.com/)
   (i) Python-dateutil (http://labix.org/python-dateutil)
   (j) Pytz (http://pytz.sourceforge.net/)
   (k) Six (http://pythonhosted.org/six/)
2. Minor modifications have been made to the matplotlib library files. To include these modifications, copy the file(s) contained in *SARDM\v0.6\modified-lib-files* to the appropriate library directory within the Python installation (such as within: *C:\Python27\Lib\site-packages\matplotlib*). It is recommended to back-up the original files prior to over-writing them with the modified files.
3. Set the PythonPath environment variable to include the tool's "lib" directory. To do this, go to Control Panel > System (Properties) > Advanced (tab) > Environment Variables (button). Under "user variables…" edit or create a path called "PYTHONPATH" with value (separate from any existing entries using ;) *C:\[your path]\ SARDM\v0.6\lib*
4. The tool can be then run using: *SARDM_v0_5.py*

1.3 Configuring the toolset

Configure the application via the text file: *SARDM_config.txt*. After the running SARDM for the first time (either via the Windows installation or via Python), the configuration file is copied to the user application directory (such as the Windows directory: *C:\Documents and Settings\Guest\Application Data\SARDM v0.6*)
Currently configuration includes:

- All options saved via the SARDM Options menu, including the disk location of the RAMAS Metapop executable, the default working directory for the tool, and various other default settings for the tool. Although it is possible to alter these via the configuration file, it is recommended to utilise the tool's Option menu for these settings.
- The configuration file also allows the user to customise the order that the parameters appear within the tool (via *parameters = […]*), and the list of probability LHS distributions that are utilised to sample each parameter: *parameter_includes_lhs_distributions['*parameter*'] = [*list of distributions*]*
  where the list of distributions are a subset of:
  ['Uniform', 'Gaussian', 'Triangular', 'Lognormal', 'Beta']
  The first distribution in the list is the default when a parameter is selected.

Other configuration is currently hard-coded into the application and can only be adjusted if the application is run as a Python program. Much of the complexity of SARDM's operation, which relates specifically to integrating with RAMAS Metapop files, is built into its configuration, thereby facilitating the tool to be extended and re-configured to operate with other simulation environments in future releases.

2 User Guide

Run the tool (*SARDM_v0_5* executable or Python program):

1. MP Sampling Generation (main window)
   Create the modified RAMAS Metapop input (MP) files using the desired sampling method.
2. RAMAS Metapop Batch (via Run menu)
   Automatically created batch file runs the RAMAS Metapop simulations for each modified MP file.
3. MP Results Summary Generation (via Run menu)
   Create result summaries using the RAMAS Metapop simulation result files.

The RAMAS Metapop batch and results summary (steps 2 & 3) can be run automatically via the option on the main window.

2.1 MP Sampling Generation

To generate modified MP files containing sampled parameters for sensitivity analysis follow the 4 step procedure:

1. Load the baseline input file. Select and load an input file. Currently the file must be compatible with the tool's configuration (default is a RAMAS Metapop MP input file). This may be adjusted within the main program if the user is running the tool as a Python program. It is anticipated that future versions will be more extensively configurable for users running the tool as a stand-alone application.

2. Set the sampling:
   (a) Establish the type of sampling that should be implemented. Currently the choices are Latin Hypercube, Random and Full factorial.
   (b) If the sampling type is Latin Hypercube or Random set how many times the parameters should be sampled between the upper and lower bounds.

3. Select the parameters and settings:
   (a) Determine which parameters should be modified by selecting checkboxes. Currently the configured possibilities are:
      - Initial Abundance
      - Carrying Capacity
      - Fecundity Rates
      - Survival Rates
      - Environmental Variation
      - Rmax
      - Allee Effect
      - Dispersal Matrix
      - Correlation Matrix
      - Probability of a Catastrophe 1
      - Stage Multiplier 1
      - Local Multiplier 1
      - Probability of a Catastrophe 2
      - Stage Multiplier 2

- Local Multiplier 2

Note: The checkboxes for the parameters associated with an optional second catastrophe (probability and local & stage multipliers) are only enabled if it is defined. Likewise, other parameters are only enabled if the relevant options are set in the input (MP) file (as opposed to the functional definition only).

(b) When selected the values of each parameter loaded from the baseline MP file may be viewed via the [v] button. Individual values may be selected so as to be utilised when the 'Set via' field is set to 'Selected value'.

(c) If the type of sampling is Latin Hypercube, then the sampling distribution may be selected from:
   a. Uniform distribution
   b. Gaussian distribution
   c. Triangular distribution
   d. Lognormal distribution
   e. Beta distribution

   Each parameter may be configured to utilise a subset of these distributions. Any distributions not included for a given parameter will appear in the list but will not be selectable. The first distribution listed for each parameter within the configuration file will become the default when each parameter is selected.

(d) The distribution settings may be 'Set via' a percentage of the baseline parameter values or via the value selected via the 'Loaded values' in (b).

(e) The parameter settings will vary dependent on the sampling type selected:
   i) For Latin Hypercube sampling from the uniform distribution, set the lower and upper bounds for the distribution as a percentage of the baseline parameter values (eg. lower = 80%, upper = 120%).
   ii) For Latin Hypercube sampling from the Gaussian (Normal) distribution, set the mean and standard deviation for the distribution as a percentage of the baseline parameter values (eg. mean = 100%, std. dev. = 10%).
   iii) For Latin Hypercube sampling from the Triangular distribution, set the lower bound (a), upper bound (b) and mode (c) for the distribution as a percentage of the baseline parameter values (eg. a = 80%, b = 120%, c = 110%)
   iv) For Latin Hypercube sampling from the Lognormal distribution, set the lower bound and scale for the distribution as a percentage of the baseline parameter values (eg. lower = 80%, scale = 5%), and the shaping parameter sigma as a positive decimal number (eg. sigma = 0.5).
   v) For Latin Hypercube sampling from the Beta distribution, set the lower and upper bounds for the distribution as a percentage of the baseline parameter values (eg. lower = 80%, upper = 120%), and the shaping parameters alpha and beta as a positive decimal numbers (eg. alpha = 0.5, beta = 2.0).
   vi) For Random sampling or Full factorial set the bounds between which each selected parameter is to be sampled (eg. +/- 10%)

(f) If the type of sampling is Latin Hypercube, then a plot of the distribution may be viewed.

4. Sampling file generation. The tool produces four outputs:
   (a) Modified .MP files (the number coinciding with the number samples or the full factorial combination). The replications and simulation duration may also

be optionally modified within each MP file generated. These settings are available via the Options menu.

(b) A data frame (*data_frame.txt*) that lists all the modified values (or a percentage for matrices with more than one value) for each file. Files are labelled so that they can be linked to the data frame. Parameters that are outside of expected values, as defined by configured constraints, are indicated via a '*', and explanations are provided at the end of the listing.

(c) A MP batch file (*mp_batch.bat*) for running RAMAS Metapop with the generated MP files.

(d) A data file (*generation_data.dat*) containing the details of the sampling file generation. This file is utilised when running the MP Results Summary.

The generated files are placed within the selected directory. If the selected directory is not empty a sub-directory is created and named using a current date and time stamp (to the nearest millisecond).

The tool utilises a workflow that ensures the user has completed the first three steps before sampling file generation can occur.

2.2 RAMAS Metapop Batch

The generated RAMAS Metapop batch file (*mp_batch.bat*) may be run via the Run menu, automatically via the selection in Step 4 of the RAMAS main window, or via double-clicking on the batch file in Windows Explorer. The batch file runs RAMAS Metapop simulations for each modified MP file generated, and appropriate results files are generated within the same directory as the generated MP files.

2.3 MP Results Summary Generation

This functionality loads the results of the batched RAMAS Metapop runs corresponding to MP files generated via the MP Sampling Generation Tool, and generates files summarising key selected results, including some general linear model (GLM) regression analysis. The original inputs (modified MP parameters) and outputs (corresponding results) are also provided in both text and comma-separated values (CSV) formats. The tool also generates series plots and text and CSV data for the predicted Abundance and Metapopulation Occupancy including:

- mean
- mean ± 1 S.D.
- mean ± 2 S.D.
- minimum-mean-maximum

To generate a summary of RAMAS Metapop batch run results, corresponding to MP files generated via the MP Sampling Generation Tool, follow the 3 step procedure:

1. Load the results directory. Select the directory containing the RAMAS Metapop batch run results. This directory must have been generated using the MP Sampling Generation Tool and should contain the data file: *generation_data.dat*.

2. Select the results that are to be collected for the result summary. Currently the configured possibilities are:
   - Expected Minimum Abundance

- Final number of occupied patches
- Final N for persistent runs
- Total extinction risk
- Quasi extinction risk

3. Generate the result summary files. The tool produces output files:
   (a) The original inputs (modified MP parameters) in text and CSV formatted files: *result_inputs.txt* and *result_inputs.csv*.
   (b) The outputs (corresponding results) in text and CSV formatted files: *result_outputs.txt* and *result_outputs.csv*.
   (c) The result summary including some general linear model (GLM) regression analysis in file: *result_summary.txt*.
   (d) A text file: *keys.txt*; describing each abbreviated input and output parameter name.
   (e) Series plots text and CSV data files for the minimum, mean, mean ± 1/2 S.D., and maximum predicted Abundance and Metapopulation Occupancy: *Predicted_Abundance.[txt/csv]* and *Predicted_Metapopulation_Occupancy.[txt/csv]*.

4. Once the result files have been generated, the series plots for the mean, mean ± 1 S.D., mean ± 2 S.D., and minimum-mean-maximum predicted Abundance and Metapopulation Occupancy may also be viewed via the selection that becomes available within the Results Generation window.

The results summary generation may run automatically (with all results selected) via the selection in Step 4 of the RAMAS main window.

2.4 SARDM Options

Various tool configurations may be saved via the Options menu. Currently the following options are available:

1. General file options, including:
   (a) RAMAS Metapop program file (exe) location.
   (b) Default working directory.

2. MP sampling generation options, including:
   (a) Option for resetting sampling parameters when a new baseline MP file is loaded.
   (b) Default numbers of samples for LHS and random sampling.
   (c) Option for a minimum number of samples.
   (d) Default LHS settings.
   (e) Default sample bounds for random and full-factorial sampling.
   (f) Optional setting of RAMAS Metapop iterations per scenario (replications) and simulation duration in the generated MP files.

3. Option of automatically running RAMAS Metapop batch and result summary generation.

3 Testing

The SARDM tool is currently modularised into 5 parts:

1. Main program (*SARDM_v0_5.py*): containing the tool's graphical user interface (GUI) and tool workflows.
2. MP file helper module (*MpFileExtractorAndGeneratorHelper.py*): containing functionality for: extracting parameter values from baseline MP files, and results from RAMAS Metapop result files; generating modified MP files, data frame files, generation data files, RAMAS Metapop batch files, and result summary files.
3. Sampling generation helper module (*SampleGenerator.py*): containing functionality for generating sampling multipliers for all three sampling methods, and performing the multiplication in order to modify parameter values.
4. Bound constraint helper module (*MpParameterBoundConstraintHelper.py*) for calculating maximum bounds, or minimum and maximum percentage, that can be applied to sampling, and for generating appropriate warnings within the GUI and data frame files.
5. Population matrix generator module (*PopulationMatrixGenerator.py*) for generating population migration/dispersal and correlation matrices from parameterised distance-based functions.

The toolset's functionality may more easily be checked for validity by decomposing testing into unit tests for individual modules and the functions within them.

3.1 Main program tests

Testing the main program involved running the tool and testing the GUI and checking the file generation. Much of this was performed during development and includes tests to:

1. Ensure the GUI functions in a usable manner for each step in the tool's workflow for:

   MP Sampling Generation

   (a) File load brings up file dialog that searches for MP files. When loaded an indication of the loaded file name is provided.
   (b) Selecting/deselecting sampling type enables/disables the number of sample entry fields (where required). Only positive integers are allowed to be entered (restricted key input) into these fields. The parameter settings section changes dynamically when the sampling type is selected so as to reflect the appropriate settings (i.e. mean and std. dev. for Latin Hypercube Gaussian; a, b, and c for Triangular; and bound for all others).
   (c) Selecting/deselecting parameters enables/disables the parameter settings entry fields. Only non-negative floats are allowed to be entered (restricted key input) into these fields.
   (d) The "Generate Files" button only becomes enabled when steps 1, 2, and 3 are complete (including valid sample bound and number of samples where applicable). An indication of the steps that need to be completed is maintained beside the button. When complete, the button becomes enabled and an indication of readiness to generate files is provided. When the enabled

"Generate Files" button is pressed the MP files and data frame are generated into the date-time stamped directory. Once complete, an indication of the number of files generated and the directory name are provided. The tool then becomes ready to generate more sampling files as required, providing all steps remain complete. If any errors occur during the generation process an appropriate error message will appear and the file generation will not occur. This can be tested by loading a file that is not compatible with the tool configuration.

RAMAS Metapop Batch

Selecting the RAMAS Metapop batch from the Run menu should run the batch file, evident via the resulting command and RAMAS windows.

MP Results Summary Tool

(a) Result load brings up a directory selection dialog. Appropriate warnings are given when expected files are missing. When the results are successfully loaded an indication of the loaded directory name is provided, and the result selection boxes are enabled.
(b) Selecting one or more results enables the result summary generation.
(c) The "Generate Summary" button only becomes enabled when steps 1, and 2 are complete. An indication of the steps that need to be completed is maintained beside the button. When complete, the button becomes enabled and an indication of readiness to generate files is provided. When the enabled button is pressed the result summary is generated into the same directory as the result files. Once complete, an indication of the directory name is provided. The tool then becomes ready to generate more result summaries as required, providing all steps remain complete. If any errors occur during the generation process an appropriate error message will appear and the generation will not occur.
(d) Once the result summary files have been generated, the tool should display a selection of available plots (series plots for the mean, mean ± 1 S.D., mean ± 2 S.D., and minimum-mean-maximum predicted Abundance and Metapopulation Occupancy). Each selected plots should open in another window when the 'View' button is pressed.

SARDM Options

(a) Option settings should save to the configuration file. This is evident via inspecting the configuration file located in the user's application data directory, for example:
C:\Documents and Settings\Guest\Application Data\SARDM v0.6
Successful option saving is also evident when the options window is inspected after closing and re-opening SARDM.
(b) Once saved, current option settings are also applied elsewhere where appropriate within the SARDM tool.

2. Checking each tool's generated files may be performed manually:

MP Sampling Generation

(a) The generated MP and data frame files may be checked by cross-referencing the values in the MP files with the data frame values. As the baseline files are

relatively complex. Validating the tool's operation can also be achieved via testing the helper modules with simpler examples.

(b) The generated RAMAS Metapop batch file may be tested by running it. This should generate result files with corresponding sample/run numbers (eg. Abund_0001.txt).

MP Results Summary

(a) The generated result summary file may be checked by cross-referencing the parameter values with the data frame, and the result values with the appropriate result files (as generated by the RAMAS Metapop batch file). Validating the tool's operation can also be achieved via testing the helper modules with known examples.

3.2 MP file helper tests

Tests for validating the functionality of the MP file helper module are provided within "*Test\MpFileExtractorAndGeneratorHelperTest.py*". This Python program runs using: simple examples, including a simple MP baseline file "*Test\test_1_baseline.mp*" and matching configuration; and known examples, including a realistic MP file, RAMAS Metapop batch file, and result files generated via a batch file. The tests include:

1. MP file extraction and generation.
   Part I: Using a simple example MP file. Check that:
   (a) The parameter values are correctly extracted from the baseline mp file.
   (b) The template is correctly generated.
   (c) The replacement values are substituted correctly into the modified mp file.
   (d) The output formats are as per configuration.
   Part II: Using a realistic example MP files, including Abalone, GCB, Orangutan, and Turtle case studies. Check that:
   (a) Test the resolution of dynamic MP parameter mapping configuration.
   (b) The parameter values are correctly extracted from the baseline mp file.
   (c) Matrices are generated from extracted function parameters where appropriate.
   (d) The template is correctly generated including the modification of additional settings.
   (e) The replacement values are substituted correctly into the modified mp file.
   These examples test various combinations of alternative or conditional extraction strategies, including:
   - Conditional inclusion of Rmax, Carrying Capacity and Allee Effect dependent on density dependence type, which may be defined for each population or common to all.
   - The extraction of multiple matrices (or layers) for Fecundity Rates, Survival Rates and Environmental Variation.
   - The conditional extraction or function-based generation of matrices for Dispersal Matrix and Correlation Matrix.
   - The conditional inclusion of Dispersal Matrix and Correlation Matrix when populations > 1.
   - The conditional inclusion of Probability of a Catastrophe, Local Multiplier, and Stage Multipliers (1 and 2) dependent on catastrophe Extent (Local/Correlated/Regional) and Affects (Abundances/Vital Rates/Carrying Capacities/Dispersal Rates).

- The conditional extraction of Probability of a Catastrophe and Stage Multipliers (1 and 2) dependent on Extent and Affects.
- The conditional extraction of the temporal trend and other filenames required, including: user defined functions (.DLL); relative fecundity (.FCH); relative survival (.SCH); relative dispersal (.DCH); temporal trend in K (.KCH); relative variability-fecundity (.VCH); relative variability-survival (.VCH); catastrophe probability (.PCH); catastrophe local multiplier (.MCH); and map features (.MAP).

(f) The temporal trend and other linked files are appropriately modified, indexed, and generated or copied into the generation directory.

2. Data frame generation. Check that:
   (a) Data frame headings and entries are correctly generated.
   (b) Entries are in the correct format and appropriately presented as a value or %.
   (c) Constraint violations are marked and noted.
3. RAMAS Metapop batch generation. Check that:
   (a) Batch entries are correctly generated.
   (b) The MP files are copied into 'originals' directory.
   (c) Batch file runs correctly (calls RAMAS Metapop) and generates numbered output files.
4. Save and load generation data. Check that:
   (a) Complex data (containing dictionaries, lists, and numpy arrays) saves to file as a serialised object.
   (b) Complex data loads from file in its original form
5. Load RAMAS Metapop Results. Check that:
   (a) Collects an ordered list of the required result files in their generic form.
   (b) Appropriate warning when *generation_data.dat* missing.
   (c) Appropriate warning when batch results missing.

   (d) Result extraction:
      i) Result components are correctly extracted from result files.
      ii) Results are correctly calculated from result components.
      iii) Results are correctly extracted from multi-run result files.
   (e) Appropriate warning when batch results incomplete.
   (f) No warning when successfully loaded results.
   (g) Handle division by zero if total extinction risk = 1 when calculating final N for persistent runs.
   (h) Handle case when the first threshold value > 0 in IntExtRisk result file (total extinction risk = 0).
   (i) Handle other quasi-extinction risk extraction cases:
      i) Quasi-extinction threshold falls below minimum threshold value in IntExtRisk result file (quasi-ext-risk = 0).
      ii) Quasi-extinction threshold falls above maximum threshold value in IntExtRisk result file (quasi-ext-risk = 1).
      iii) Quasi-extinction threshold falls between threshold values in IntExtRisk result file (linear interpolate).
6. Generate Result Summary. Check that:
   (a) Complete batch and full results produces correct summary file.
   (b) Incomplete batch and partial results produces correct summary file.
7. Generate Result Summary (including regression model). Check that:

(a) Perfect separation error occurs with several input parameters when sample number is small.
(b) Generalised Linear Models run correctly on other occasions.

8. Generate Result Plots Data (including files). Check that:
   (a) Result plot data is extracted from result files.
   (b) Result plot data files are generated.

3.3 Sampling generation helper tests

Tests for validating the functionality of the sampling generation helper module are provided within "*Test\SampleGeneratorTest.py*". This Python program runs the following tests using a simple example:

1. Test Latin Hypercube sampling multiplier generation for:
   (a) Uniform distribution.
   (b) Normal (Gaussian) distribution
   (c) Triangular distribution
   (d) Lognormal distribution
   (e) Beta distribution
   (f) Mixed distributions
2. Test Random sampling multiplier generation
3. Test Full Factorial multiplier generation
4. Test lower and upper threshold functions for unbounded distributions
5. Test multiplier to modify parameter values
6. Graphical plotting tests for Latin Hypercube Sampling distributions (depends on the matplotlib module)
7. Graphical plotting verification tests for Latin Hypercube Sampling Stratified Generation (depends on the matplotlib module)
8. Graphical plotting verification tests for distribution plot values generation (depends on the matplotlib module)

3.4 MP parameter bound constraint helper tests

Tests for validating the functionality of the MP parameter bound constraint helper module are provided within "*Test\MpParameterBoundConstraintHelperTest.py*". This Python program runs the following tests using a simple example:

1. Single value matrices:
   (a) No constraints defined.
   (b) Lower value constraint only.
   (c) Lower and upper value constraints.
   (d) Upper value constraint applied to a value that already exceeds it
2. Single row matrix:
   (a) Lower and upper value constraints.
3. Single column matrices:
   (a) Lower and upper value constraints.
   (b) Upper sum constraint (implicitly all rows).
   (c) Lower and upper value constraints plus upper sum constraint from specified row.
4. Matrix:
   (a) Lower and upper value constraints plus upper sum constraint from specified row.

(b) Lower and upper value constraints plus upper sum constraint applied to a sum that already exceeds it.
(c) Lower and upper value constraints plus upper sum constraint applied to matrix with zero and small entries.
(d) Upper value constraint plus upper sum constraint applied to matrix with zero and small entries.

5. Check Modified Matrix Constraint Violations:
   (a) No constraints registered for parameter.
   (b) No violations for registered parameter.
   (c) Lower & upper value plus column sum violations.
6. Check constraints and violations with multiple layered matrices:
   (a) Lower cell constraint.
   (b) Upper cell constraint.
   (c) Column sum constraint.
   (d) Lower & upper value plus column sum violations.

3.5 Population matrix generator tests

Tests for validating the functionality of the population matrix generator module are provided within "*Test\PopulationMatrixGeneratorTest.py*". This Python program runs the following tests using a simple example:

1. Calculate inter-population distances using values from a sample RAMAS Metapop input (MP) file.
2. Calculate dispersal/migration matrix Mij.
3. Calculate correlation matrix Cij.

4 Known Issues or Limitations

In the first version of the tool there are several known issues and limitations that may be addressed in later versions if necessary.

4.1 Extraction and generation errors

Errors associated with parameter extraction, sampling generation, and file generation are presently only captured within the main program as error dialogs, and may lack specific information about what has caused the error.

4.2 Configuration

Presently, the majority of the configuration is hard-coded into the main program and can only be altered by a user running the tool as a Python program. No complex cross-checking of the compatibility of a loaded baseline MP file with the parameter configuration is made. Error dialogs (as per 4.1) may indicate that the loaded file is incompatible with the configuration.