NISTIR 7739

Open Monte Carlo Engine User Manual

Ruediger Kessel

National Institute of Standards and Technology
U.S. Department of Commerce

NISTIR 7739

Open Monte Carlo Engine User Manual

Ruediger Kessel
Information Technology Laboratory
Applied and Computational Mathematics Divisiont

November 2010

U.S. Department of Commerce
Gary Locke, Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, Director

Open Monte Carlo Engine
User Manual
[OMCE V: 1.2.14.6]

Rüdiger Kessel (ruediger.kessel@gmail.com)
National Institute of Standards and Technology,
Gaithersburg, MD 20899, USA

Contents

1 Introduction

The Open Monte Carlo Engine (`OMCE`) is a full featured command line Monte Carlo engine. It reads a model description in xml-format and runs a Monte Carlo simulation to calculate a histogram, the best estimate (mean) and the standard uncertainty. In Addition, probabilistically symmetric or shortest coverage intervals are calculated. The `OMCE` can be used as an alternative method to the mainstream GUM [1] to evaluate the measurement uncertainty. The `OMCE` implements the Monte Carlo integration method as described in the Guide to the Expression of Uncertainty (GUM) Supplement 1 [2]. The `OMCE` is written in Python and is public domain (open source code).

1.1 Theory of operation

The `OMCE` reads the model description in xml-format and creates an internal list of quantities, equations and results. It then evaluates the execution order for the equation system. A production and consumer pipeline is set up with two tasks. One task calculates a block of results and analyzes it, and the other task writes the block to a binary file and frees the memory. The pipeline stops when sufficient data are produced and completely written to the binary file. The last step is to write the histogram to a text file.

1.2 Limits and restrictions

The `OMCE` has the following limits and restrictions:

- The number of quantities, equations and results is not limited by the software. In practice the computer hardware, the memory and the computational power will limit the size of the model that can be simulated.
- Only explicit model equations can be simulated. `OMCE` does not include an equation solver.

1.3 Distribution of the `OMCE`

The `OMCE` is distributed as source code accompanied by Windows executable files (`OMCE.exe`, `OMCEclient.exe` and `OMCEview.exe`) for the convenience of users who do not have Python [3] installed. The current version of `OMCE` was developed with Python 2.5. Some additional packages are needed (see Section 13) to run the source code version of `OMCE` (`OMCE.py`).

1.4 Copyright and trademarks

The `OMCE` is free software with no copyright associated with it. It can be freely copied, modified and used, but it cannot be copyrighted. The idea is to have a Monte Carlo engine that can be freely studied and modified to user needs.

All trademarks mentioned herein belog to their respective owners.

1.5 Disclaimer

This software was developed at the National Institute of Standards and Technology by employees of the Federal Government in the course of their official duties. Pursuant to title 17 Section 105 of the United States Code this software is not subject to copyright protection and is in the public domain. This software is experimental. NIST assumes no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic. We would appreciate acknowledgement if the software is used.

2 Installing OMCE

OMCE is distributed as a zip-file (OMCE-1.2.14.6.zip) containing all necessary files to run it on a Windows platform without Python. The zip-file should be unpacked in a separate directory OMCE\. The OMCE can be executed directly from the command line if the OMCE\bin\ subdirectory is added to the search path. The executable version (exe) contains all packages needed to run it. In addition, the source code is included. Linux users should use the source code version. The OMCE source code makes use of Python 2.7 [3] and several non-standard open source Python packages (see Section 13) imported in the beginning of OMCE.py. The packages must be installed before the source code version can be run.

Contents of OMCE-1.2.14.6.zip		
File	Location	Description
OMCE.exe	OMCE\bin\	Exe version of the OMCE
OMCE.cfg	OMCE\bin\	default option configuration file
OMCE.xsd	OMCE\bin\	xml schema definition of the OMCE format
OMCE.ofd	OMCE\bin\	OMCE statistical output file definition
OMCE.vfd	OMCE\bin\	OMCE var file definition
OMCEclient.exe	OMCE\bin\	Exe version of the OMCE
OMCEclient.cfg	OMCE\bin\	client option configuration file
ALT.ofd	OMCE\bin\	alternative statistical output file definition
ALT.vfd	OMCE\bin\	alternative var file definition
.	OMCE\bin\	python runtime environment
OMCE.pdf	OMCE\doc\	OMCE user manual
OMCE\Schema.html	OMCE\doc\	OMCE xml schema description
OMCE.py	OMCE\src\	OMCE simulator (Python source)
OMCE.cfg	OMCE\src\	default option configuration file
OMCE.xsd	OMCE\src\	xml schema definition of the OMCE format
OMCE.ofd	OMCE\src\	OMCE statistical output file definition
OMCE.vfd	OMCE\src\	OMCE var file definition
OMCEbase.py	OMCE\src\	OMCE base module (Python source)
OMCEclient.py	OMCE\src\	OMCE client (Python source)
OMCEclient.cfg	OMCE\src\	client option configuration file
OMCEd.py	OMCE\src\	init.d compatible daemon wrapper
OMCEd_config	OMCE\src\	config file for the daemon wrapper
ALT.ofd	OMCE\src\	alternative statistical output file definition
ALT.vfd	OMCE\src\	alternative var file definition
OMCEview.py	OMCE\src\	histogram viewer OMCEview
OMCEview.cfg	OMCE\src\	OMCEview configuration file
OMCEview.exe	OMCE\viewer\	EXE-version of the histogram viewer OMCEview
OMCEview.cfg	OMCE\viewer\	OMCEview configuration file
.	OMCE\viewer\	python run-time environment for OMCEview
*.omc	OMCE\examples\	example xml files (OMCE format)
*.xml	OMCE\examples\	example xml files (alternative format)
!run_all_OMCE.bat	OMCE\examples\	batch file to run all examples in OMCE format
!run_all_ALT.bat	OMCE\examples\	batch file to run all examples in alternative format
*.ohd	OMCE\examples\ref\	reference results for the examples in OMCE format (-seed=1)
*.sta	OMCE\examples\ref\	reference results for the examples in alternative format (-seed=1)

After the installation, the windows batch file !run_all_OMCE.bat or !run_all_ALT.bat in OMCE\examples\ should be run to verify the installation. This will run the example xml files and compare the results with the reference results stored in OMCE\examples\ref\. The random number generator is seeded for these runs so the results must exactly match the reference results.

3 Running OMCE

The OMCE is a command line tool which is designed to be used in batch scripts or to be called by other programs. If the directory of OMCE (exe-version) is included in the path, it can be executed with the following command:

```
OMCE filename.omc
```

This will start the OMCE, loading filename.omc and creating a file named filename.ohd. All options have default values. If no file name is given, a list of options with their default values and the implemented error codes is printed.

The general format for using the OMCE is:

```
OMCE filename.omc|filename.xml [options] [parameters]
```

The input file should be either in omc-format or in the alternative xml-format. See option -fi and Section 4.1 on file formats for more details.

All option follow the following format:

```
-option=value
```

The option name option is string of one to four characters identifying the option and is always followed by an equal sign. The format of the value depends on the option and can be either an integer, a floating point number or a string. See Section 3.1 for a detailed discussion.

Parameters have the following format:

```
name=value
```

The parameter names are defined in the input omc-file (see Section 4.1.1). The value must be a floating point number.

Note: The name of the input file must be specified before any parameters.

If there is an error, OMCE will terminate with a non-zero exit code. See Appendix A for a list of possible error messages and exit codes.

3.1 Options

The options can be given in any order but they should be separated by a space and should be given after the omc-file name.

Note: Additional options starting with a double dash (-) are supported for the server mode as discribed in Section 11.

-a This switch activates the generation of the statistical data file (.ohd) and is set to 1 by default. This switch should be set to zero if the internal data analyzer is not used and the binary data file (-wb=1 or -wb=2) is analyzed by some external tool.

-ac This option controls the correlation analysis of the simulated output data.

-ac=	correlation analysis
0	no correlation analysis
1	between results (default)
2	results and their inputs
3	all

-ad This option sets the adaptive mode tolerance divisor to the value given (see section on adaptive mode). The tolerance is calculated based on the standard uncertainty or the probability interval and the number of significant digits divided by the tolerance divisor. The default value is 1.0.

The adaptive mode is controlled by the following options: -ad, -af, -am, -ap, -at, -k, -mr, -sd.

`-af` This option sets the probability level for the decisions during the adaptive mode. If option `-k=0` or if the alternate algorithms are used (`-am` > 0) a t-factor is calculated based on the probability controlled by this option and the number of simulation runs so far.

The adaptive mode is controlled by the following options: `-ad`, `-af`, `-am`, `-ap`, `-at`, `-k`, `-mr`, `-sd`.

`-am` This option controls the method how the number of runs are determined during the adaptive mode. The default `-am=0` uses the sequential checking scheme defined in GUM Supplement 1. Alternatively, `-am` > 0 activates alternative 2-stage schemes to predict the total number of runs.

`-am=`	adaptive algorithm used during simulation
0	GUM Supplement 1 based algorithm
1	modified Stein 2-Stage scheme (see [4]) with S1 accuracy definition
2	modified Stein 2-Stage scheme (see [4]) with fixed relative accuracy definition
3	prediction of the needed runs with S1 accuracy definition
4	prediction of the needed runs with fixed relative accuracy definition

The adaptive mode is controlled by the following options: `-ad`, `-af`, `-am`, `-ap`, `-at`, `-k`, `-mr`, `-sd`.

`-ap` This option sets the probability of the interval which is used for the tolerance check in the adaptive mode. This interval is calculated for every block of data. If bit 2 of the option `-at` is set then the standard deviation of the average of all calculated intervals is multiplied by the k-factor (option `-k`) and compared with the tolerance based on option `-sd` and `-ad`.

The adaptive mode is controlled by the following options: `-ad`, `-af`, `-am`, `-ap`, `-at`, `-k`, `-mr`, `-sd`.

`-at` This option controls in detail which parameter is used to for the check in the adaptive mode (see Section 6). The value of this option is a bit combination which enables the different checks The default is to the check the standard deviation of the mean, the standard uncertainty and the limits.

`-at=`	parameter used during the check
1	standard deviation of the means
2	standard deviation of the uncertainty
3	standard deviation of the means and the uncertainty (see [2] Section 7.9.4 Note 5)
7	standard deviation of the means, the uncertainty and the limits (default see [2] Section 7.9.4)

`-bs` The internal data processing in the `OMCE` operates on data blocks with a fixed block size. The block size has influence on the performance and the statistical properties of the result. Usually the default block size (10000 values) is a good compromise between performance and statistical rigor. The block size can be changed for experimental purposes or if the value of option `-nl` or `-nc` is reduced.

In the adaptive mode the block size is adjusted automatically and the value given with this option will be overridden.

`-cfg` This option sets the name of a configuration file which contains default options in separate lines. The default value is `-cfg=OMCE.cfg`.

`-cl` This option controls the lower limit of the absolute value of the calculated correlation coefficient. The default value is 0.001.

`-da` By default (`-da=0`) an existing output file will be replaced by new simulation data. If this option is set to `-da=1`, no header is printed to the file and the data is appended. If the data should be distinguishable from prior existing data, set the option `-sid` to a different start value. Warning: The format of the existing output file will not be checked.

`-de` This option controls how the name of the output files is formed from the given input filename. If this option is set (default), the extension (.omc) is deleted from the filename before the output file extension is added; otherwise the extension is added without deletion. Warning: any existing output file (.dat, .bin, .ohd, .sta) will be overwritten without warning.

`-e1` The `OMCE` will automatically try to correct for small negative eigenvalues in the correlation matrix. This option controls the limit for the correction. The value must be negative and the default is -0.1.

`-e2` When the correlation matrix needs to be corrected, the Least Maximum Norm of the correction is calculated. This option controls the limit for this norm. The default value is 0.01.

`-fi` The `OMCE` supports two input formats which are both based on xml. The native `OMCE` format allows a simple description of the evaluation model. The details of the format can be found in `OMCE_schema.html`. As an alternative, `OMCE` can read an alternative xml-format.

`-fi=`	input format
0	`OMCE` format (default)
1	alternative xml-format

`-fo` As a default, `OMCE` forms the output file name based on the given input file name. With this option an explicit output file name can be given.

`-hc` The resolution of the histogram is set by this option (number of bars). The value should be between 1 and 100000. The default is 200 bars.

`-hf` The width of the histogram is extended by the factor given with this option. The value should be between 0.1 and 100.0. The default is 1.2.

`-hp` The width of the histogram is controlled by a probability interval to capture the relevant part of the distribution. The probability of the interval should be given as a value between 0.001 and 1. A value of 1 creates an interval that contains all values calculated during the simulation. The default is 0.99 (99 %).

`-i` This options controls the interpretation of the coverage intervals. By default (`-i=0`), all coverage intervals are probabilistically symmetric. If the option is non-zero then the shortest coverage intervals are calculated.

Note: In case the result distribution has not a single mode then the shortest coverage interval is ambiguous. The algorithm proposed by GUM Supplement 1 and which is implemented with `OMCE` is not detecting this case. As a consequence the reported coverage interval might not be reliable in this case and this option should not be used.

Probabilistically symmetric intervals are unambiguous and can therefore be used in all cases.

`-info` If the info option is non zero, `OMCE` prints details about the xml input file or some internal tables and quits.

`-info=`	description
0	(no information)
1	details of the model given in the xml-file: input quantities equations execution order results
2	list of predefined symbols and functions
3	list of symbols usable in format definition files, array symbols are followed by []
4	list of reserved words not to be used as quantity names
5	list of error messages in LaTeX format

`-io` This option controls the sampling for imported quantities (see element `<Import>` in Section 4.1.9). If `-io=0` (default) then the binary data is sampled at random from binary files. If `-io=1` then the data is sampled sequentially from the binary file in the same (reproducible) order as it was written. If the number of runs (`-mcs`) is equal to the number of values in the binary file then the exact same sequence for the quantities is reproduced.

`-k` The k-factor scales the tolerance check during the adaptive mode. The default value is 2.0. A larger value will require more Monte Carlo runs. If this option is set to zero, the k-factor will be evaluated from a t-table (see option `-ap`) look up based on the number of runs so far.

The adaptive mode is controlled by the following options: `-ad`, `-af`, `-am`, `-ap`, `-at`, `-k`, `-mr`, `-sd`.

`-l` This option allows to repeat the simulation the given number of times is it is set not equal to zero. Note: This option is only effective if the omc-file does not contain a section `<Simulations>` and will be ignored otherwise. The default value is `-l=0`.

`-lc` The correlation matrices will be printed if this option is set. The default is `-lc=0`.

`-lc=`	description
0	no matrix printed (default)
1	print input correlation matrix
2	print result correlation matrix
3	print input and result correlation matrix

`-lmn` This option defines the matrix norm type.

`-lmn=`	description
0	plain norm
1	weighted norm (default)

The weighted norm uses the larger correlation coefficient as a weight for the difference. The minimum weight is 0.1.

`-lt` If this option is set (`-lt=1`), `OMCE` prints the tolerances used in the adaptive mode together with the standard deviations evaluated (see section on adaptive mode for details).

`-mcp` This option selects the matrix correction procedure for the the correlation matrix in the event that the matrix is found to be not positive semi-definite.

`-mcp=`	description
0	spectral decomposition (default)
1	Near4 (N. J. Higham [5])
2	Random Walk

`-mcs` This option controls the number of simulation blocks. The total number of trails is the value of this option multiplied by the block size (option `-bs`). This option will override the value given in the xml-file. A value of zero or minus one activates the adaptive mode. The default value is -1.

`-mr` To prevent the simulation from running for an excessive amount of time during the adaptive mode, a maximum number of blocks (runs) can be set by this option. The value given here is multiplied by the block size (see option `-bs`). The default value is 10000 blocks.

The adaptive mode is controlled by the following options: `-ad`, `-af`, `-am`, `-ap`, `-at`, `-k`, `-mr`, `-sd`.

`-nc` This option controls the fraction of valid data that is needed to evaluate the correlation coefficients. The default value is `-nl=0.5`. The simulation is aborted if the fraction of valid data available to calculate the correlation coefficients in one data block is smaller than the value controlled by this option. Lowering this value might only be advisable if the block size is increased at the same time.

`-nl` This option controls the fraction of valid data that is needed to continue the simulation. The default value is `-nl=0.5`. The simulation is aborted if the fraction of valid data in one data block is smaller than the value controlled by this option. Lowering this value might only be advisable if the block size is increased at the same time.

`-ofd` This option controls the file name for the output format definition file. The definition file should be placed in the same directory as the `OMCE` executable or python source. The default is '`OMCE`'.

`-p1` This option controls the exact probability of the 95 % interval. The default is `-p1=0.95`.

`-p2` This option controls the exact probability of the 99 % interval. The default is `-p2=0.99`.

`-p3` This option controls the exact probability of the 99.9 % interval. The default is `-p3=0.999`.

`-pa` This option controls the exponent used for the calculation of the average of the quantiles. If a is the value set by this option then the average is calculated based on the equation

$$\bar{q} = \sqrt[a]{\frac{1}{n}\sum_{i=1}^{n} q_i^a}.$$

The default value is `-pa=2.0`

`-pdf` The data in the histogram can either be counts (`-pdf=0`) or samples of a PDF (probability density function) (`-pdf=1`). The PDF samples are useful if the data is to be compared with a given PDF. The default is `-pdf=1`.

`-po` If this option is given, `OMCE` will print a list of all effective options and quits. This option is useful for documenting and debugging the command line options.

`-pq` This option controls the quantile estimation method. The details to the estimation methods can be found in [6].

`-lmn=`	description
0	no interpolation (see [2] Section 7.7.2) (default)
1	linear interpolation at $n_\mathrm{b} \times p + 1/2$ (R-5)
2	linear interpolation at $(n_\mathrm{b} + 1/3)p + 1/3$ (R-8)
3	linear interpolation at $(n_\mathrm{b} + 1/4)p + 3/8$ (R-9)
4	linear interpolation at $(n_\mathrm{b} + 2)p + 1/2$

`-pr` The simulation is done in two steps. At the start a number of runs controlled by this option will be executed. After the first runs are completed, the binning interval and the tolerance for the adaptive mode are calculated and these values are used during the rest of the simulation runs. The default is 10 first runs.

`-py` Explicit Python code can be included in the OMC-file which allows the user to extend the possibilities of `OMCE`. In client server mode this implicates a security risk and it might be potentionally harmful to the server. This option enables all Python coding in the OMC-file. It is enabled by default. By disabling it (`-py=0`) all Python coding is disabled.

`-sbs` This option controls the sub block size for the alternate adaptive mode algorithms. The sub block mechanism is not activated by default (`-sbs=-1`). The minimum sub block size is 100. The block size must be dividable by the sub block size.

The adaptive mode is controlled by the following options: `-ad`, `-af`, `-am`, `-ap`, `-at`, `-k`, `-mr`, `-sd`.

`-sd` This option controls the number of significant digits for the adaptive mode. The default is 2 digits (`-sd=2`).

The adaptive mode is controlled by the following options: `-ad`, `-af`, `-am`, `-ap`, `-at`, `-k`, `-mr`, `-sd`.

`-seed` The seed of the random number generator can be set to a fixed value given by this option. A default value of zero will force a random seeding based on the system clock.

`-sev` This options controls the value of the smallest eigenvalue after correction. The default is 1E-7.

`-sid` This option controls the start value for the simulation ID. The default value is 0. The simulation ID will be incremented after every simulation.

`-t` The `OMCE` will measure and print the time elapsed to run a complete simulation if `-t=1`. More detailed information on the time elapsed for the omc-file reading, simulation and the ohd-file creation will be printed if `-t=2`.

`-v` This option controls the verbose level between 0 for all messages and 10 for no messages. The default value is 2.

`-v=`	description
9	error messages
8	warnings
7	startup message
6	finish message
5	simulation progress
4	timing information
3	standard information
2	default value
1	additional information and calculation details
0	everything else

Note: information requested by other options (like `-info`, etc.) will be suppressed if the verbose level is set larger than 3.

`-vfd` This option gives the file name for the format definition file of the var-file. The definition file should be placed in the same directory as the `OMCE` executable or the Python source code. The default is '`OMCE`'.

`-vm` This option activates the validation mode if it is set. During validation mode all simulation data are kept in memory and are analysed when the simulation runs are complete.

`-wb` The binary data calculated during the simulation can be written to a binary data file. This option controls the generation and the format. For the sorted format (`-wb=3`) the binary data will be sorted and made monotonic prior writing to the file. The option `-wb=3` should not be used if the data will be imported in other simulations via the element `<Import>`.

`-wb=`	format description
0	no binary data file
1	`OMCE` format
2	simple format
3	sorted `OMCE` format

The default is `-wb=0`. See section 4.2.1 on binary data format for details.

`-wv` This option enables the creation of a .var output file. By Default no .var file is generated (`-wv=0`).

`-xsd` The `OMCE` can validate the xml-input file against an xsd-file. With this option, the path to the xsd-file can be given. By default, the option is '-' and `OMCE.xsd` is used for omc-files and no validation is done for alternative xml-files. (see option `-fi`).

3.2 Example

```
OMCE S02.omc -pdf=1 -hc=400 -t=1
```

The OMCE will load the file S02.omc from the current directory and it will create the file S02.ohd containing histogram data with a resolution of 400 bins converted to a PDF sampling. The whole operation will be timed and the elapsed time will be shown.

4 OMCE file formats

The OMCE accepts two different xml-file formats discussed in the section about xml input formats. The OMCE creates data files of different types. The raw data is written to binary files and the histogram data and the other information is written to ASCII files in comma separated value format. The extension is .ohd.

4.1 Xml input file format

The native OMCE input file format (extension omc) is a simple xml-format [7]. It describes the model of evaluation and the information about the input quantities. See OMCE\doc\OMCE_Schema.html for details about the omc-file format structure. An omc-file has the following basic structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<Model
  Name="Model1"
  Options=""
  xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xs:noNamespaceSchemaLocation="OMCE.xsd">
  <Parameters>
   ...
  </Parameters>
  <Quantities>
   ...
  </Quantities>
  <Correlations>
   ...
  </Correlations>
  <Functions>
   ...
  </Functions>
  <Equations>
   ...
  </Equations>
  <Constraints>
   ...
  </Constraints>
  <Results>
   ...
  </Results>
  <Simulations>
   ...
  </Simulations>
</Model>
```

A <Model> element contains all information about the evaluation model. The Name and the Options attributes are optional. The Options attribute may contain a list of space separated command line

options. The information about the schema is optional and is not used in `OMCE`. The validation schema is controlled by the option `-xsd`.

4.1.1 Parameters

The optional section `<Parameters>` defines a list of symbolic model parameter values that can be used in the parametric specifications of the quantities. In this section the default values of the parameters are specified. A parameter specification has the following structure:

```
<Parameter Name="P" Value="1.0"/>
```

All attributes are mandatory. The attribute `Name` defines the parameter name (e.g. "P") by which the parameter is referred to in the definition of the quantities. The attribute `Value` assigns the default value to the parameter. The value of the parameter can be re-assigned by a command line parameter (see Section 3) or inside the section `<Simulations>`.

4.1.2 Quantities

The section `<Quantities>` defines a list of quantities which can be used in the model equations. A quantity has the following structure:

```
<Quantity Symbol="X" Unit="kg" Definition=""> ... </Quantity>
```

The attribute `Symbol` is mandatory and defines the symbol name (e.g. "X") by which the quantity is referred to in the equations. The attribute `Unit` is optional. A quantity element should contain one distribution element.

4.1.3 Correlations

The section `<Correlations>` is optional and defines a list of correlation coefficients which are used during the simulation runs. A correlation coefficient has the following structure:

```
<Coefficient Q1="X_1" Q2="X_2" Value="0.5" />
```

All attributes are mandatory. Attribute `Q1` and `Q2` define the quantities for the correlation coefficient and `Value` defines its value. The value must be greater than or equal to -1 and smaller than or equal to +1. All quantities where a correlation coefficient is given must be normal distributed. All correlation coefficients together must form a matrix which is positive semi-definite.

4.1.4 Functions

The section `<Functions>` is optional and defines a list of user defined functions to be used in `<Equation>` and `<Result>` elements. A function definition has the following structure:

```
<Function Symbol="f" Param="p1,p2" Global="X_1,X_2" Coding="" Using="">
...
</Function>
```

The attribute `Symbol` is mandatory and defines the function name (e.g. "f") by which the function is referred to in `<Equation>` and `<Result>` elements. The optional attribute `Param` defines a parameter list for the function. The optional attribute `Global` defines a list of quantities or equation names that can be used inside the function in addition to the given parameters. With the optional attribute `Coding` an alternative coding can be specified. The supported codings are `OMCE` and `PYTHON` If `Coding="PYTHON"` then the Python scipt can make use of other user defined functions, but they must be specified in the optional attribute `Using`.

`<Function>` should contain a valid expression or a valid Python script if coding is set to `PYTHON`.

4.1.5 Equations

The section <Equations> defines a list of equations or procedures which are evaluated during the simulation runs. A equation has the following structure:

```
<Equ Symbol="Y" Unit="kg" Definition=""> ... </Equ>
```

The attribute Symbol is mandatory and defines the symbol name (e.g. "Y") by which the equation is referred to in the result list or in other equations. Every equation must have a unique symbol name. The attribute Unit is optional. An element <Equ> should contain an expression which can be evaluated without refering to itself directly or indirectly.

As an alternative to the element <Equ> the element <Proc> can be used to define the calculation procedure. A procedure element has the following structure:

```
<Proc Symbols="Y1,Y2" Param="" Using="" Coding="" Units="kg,kg" Definitions="">
...
</Proc>
```

The attribute Symbols is mandatory and defines the symbol names (e.g. "Y1,Y2") which are evaluate by executing the procedure. The Procedure may use other symbols but it needs to specify these in the optional attribute Param. The procedure may also use build-in functions or user defined functions but they need to be specified in the optional attribute Using. The optional attributes Units and Definitions are ignored.

4.1.6 Constraints

The section <Constraints> defines a list of constraints which need to be met by the data during the simulation. A constraint has the following structure:

```
<Constraint Q="X"> ... </Constraint>
```

The attribute Q is mandatory and defines the name of the quantity (e.g. "X") referencing the data which is validated by the constraint. An element <Constraint> should contain a logical expression which can be evaluated to True or False. Logical expressions for constraints on input quantities can only refer to input quantities while logical expressions for constraints on quantities defined by equations can refer to any quantities. Quantity names referring to results cannot be used with constraints.

4.1.7 Results

The section <Results> defines a list of results which are observed during the simulation runs. A result has the following structure:

```
<Result Symbol="Y" Unit="kg" Definition="">Y</Result>
```

The attribute Symbol is mandatory and defines the symbol name (e.g. "Y") by which the result is referred to in the ohd- or var-file. The same symbol name may be used for both a result and an equation. A result element should contain a valid expression. Usually this expression contains only the name of an equation. Result symbols cannot be used in expressions for equations or results.

4.1.8 Simulations

The optional section <Simulations> provides a simple tool to define multiple simulations which are sequentially executed. If the section is not present in an omc-file, only one simulation is executed. All the result data for the simulations is written to the same file. The section <Simulations> contains a list of the elements <Simulation> and <Loop> given in any order.

The element <Simulation> defines one simulation run and it has the following structure:

```
<Simulation Name="Comment">
  <Parameter Name="P" Value="2.0"/>
  ...
</Simulation>
```

The attribute `Name` is optional and it specifies a comment that will be printed when the simulation is executed. The element `<Simulation>` can contain a list of parameter re-assignments. All parameter must be defined in the section `<Parameters>` before they can be used here.

The element `<Loop>` defines a for-loop for multiple simulations and it has the following structure:

```
<Loop Enum="i" From="1" To="5" Step="1.0" Name="Comment">
  <Parameter Name="P" Value="2.0*i"/>
  ...
</Loop>
```

The attribute `Enum` is optional and specifies a name for the enumerator. If the attribute is present the name given in `Enum` can be used to evaluate the values of the parameters.

The attributes `From` and `To` are mandatory and they define a range for the loop enumerator. The optional attribute `Step` specifies the value of the increment that is added (or subtracted) to the enumerator after every simulation. The default value for the `Step` is 1.0.

Note: The loop will always start with a value for the enumerator given in `From` independent if the value given in `To` is smaller or larger than `From`. The given sign of `Step` is ignored and the sign is adjusted so that the numerator is changed after every run in the direction of `To`.

The attribute `Name` is optional and specifies a comment that will be printed when the simulation is executed.

4.1.9 Distribution elements

A distribution element defines the information which is available about the quantity. The following distribution elements can be used inside a quantity element:

`<Constant> <Value>1.0</Value> </Constant>`
: The element `<Constant>` defines a constant quantity without uncertainty.

`<Normal> <Mean>0.0</Mean> <Sigma>1.0</Sigma> </Normal>`
: The element `<Normal>` defines a normally distributed quantity with the specified mean value and the specified standard deviation (`Sigma`).

`<Student> <Mean>0.0</Mean> <Scalefactor>1.0</Scalefactor> <Dof>5</Dof> </Student>`
: The element `<Student>` defines a student-t distributed quantity with the specified mean value and degrees of freedom (`Dof`) scaled by the specified scaling factor (`Scalefactor`).

`<Rectangle> <Mean>0.0</Mean> <Halfwidth>1.0</Halfwidth> </Rectangle>`
: The element `<Rectangle>` defines a rectangularly distributed quantity with the specified mean value and the specified half-width (semi-width).

`<Triangle> <Mean>0.0</Mean> <Halfwidth>1.0</Halfwidth> </Triangle>`
: The element `<Triangle>` defines a triangularly distributed quantity with the specified mean value and the specified half-width (semi-width).

`<Trapez> <Mean>0.0</Mean> <Halfwidth>1.0</Halfwidth> <Beta>0.5</Beta> </Trapez>`
: The element `<Trapez>` defines a trapezoidal distributed quantity with the specified mean value and the specified halfwidth (semi-width). `Beta` is the ratio between the evenly distributed part and the width of the distribution.

`<Ctrapez> <Mean>0.0</Mean><Halfwidth>1.0</Halfwidth><Beta>0.5</Beta> </Ctrapez>`

The element `<Ctrapez>` defines a iso-curvilinear trapezoidal distributed quantity [8] with the specified mean value and the specified half-width (semi-width). `Beta` is the ratio between the evenly distributed part and the width of the distribution.

`<Ushape> <Mean>0.0</Mean> <Halfwidth>1.0</Halfwidth> </Ushape>`

The element `<Ushape>` defines a U-shaped (arcus-sinus) distributed quantity with the specified mean value and the specified half-width (semi-width).

`<TypeA> <Value>1.0</Value> ... </TypeA>`

The element `<TypeA>` defines an observed quantity. It contains at least 2 `<Value>` elements.

`<Poisson> <Mean>1.0</Mean> </Poisson>`

The element `<Poisson>` defines a Poisson distributed quantity with the specified mean value.

`<Exponent> <Mean>1.0</Mean> </Exponent>`

The element `<Exponent>` defines a exponentially distributed quantity with the specified mean value.

`<Gamma> <Shape>1.0</Shape> <Scale>1.0</Scale> </Gamma>`

The element `<Gamma>` defines a Gamma distributed quantity with the specified shape and scale value.

`<Discrete> <Binfile Filename="filename.bin" Index="0" Q="Y"/> </Discrete>`

The element `<Discrete>` defines a discrete distribution based on binary data (see [2] Section 7.5.1). The element `<Discrete>` contains the mandatory element `<Binfile>` defining the binary data source. The binary data is read from a file given by the attribute `Filename`. The file should exist and it should have a valid binary format (see option `-wb`). By default the path is either relative to the current XML-file or an absolute file path. If the optional attribute `Syspath` is non-zero then the path is relative to the `OMCE` excutional file (in the `bin` or `src` directory). In server mode (see Section 11) `Syspath="1"` activates the loading of the binary file from the server file system instead of the client files system (which is the default).

Binary files can contain binary data for multiple quantities. The optional attribute `Index` defines the index of the quantity in the binary file. The first quantity is the default and has the index 0. If the binary contains a symbol table the optional attribute `Q` can specify the name of the quantity instead of its index. The attribute `Index` will be ignored if an attribute `Q` is specified.

After the element `<Binfile>` two optional elements `<Shift>` and `<Scale>` can be used to define a scale value all data is multiplied with and a shift value which is added to all (shifted) values. Multiple quantities might share the same binary data with different shift and scale values. Although the these quantities share the same data, they are uncorrelated.

`<Import> <Binfile Filename="filename.bin" Index="0" Q="Y"/> </Import>`

The element `<Import>` is similar to the element `<Discrete>`. It contains only the element `<Binfile>`. No other optional elements are allowed. See the paragraph above about `<Discrete>` for details about the element `<Binfile>`. As long as only one quantity is imported per file there is virtually no difference between element `<Discrete>` and element `<Import>`. The user might use the the element `<Discrete>` if the distribution needs to be scaled or shifted.

If multiple quantities are imported from the same file with the element `<Import>` then the dependency structure (correlation) present in the binary data will be maintained in the current simulation by sampling jointly for all quantities from this file. Option `-io` controls if a random sampling (`-io=0`) or a sequential sampling (`-io=1`) is used.

The same binary file can be used for a `<Discrete>` element and for a `<Import>` element by using the same filenames and indices. The quantities share the distribution but they are sampled independently (not correlated).

4.1.10 Expressions and equations

The elements `<Equ>`, `<Function>` and `<Result>` require an expression as their content. The format of the expression is basically the format of a Python expression. The operator `+`, `-`, `*`, `/` and `**` can be used in expressions together with predefined functions, constants and the parentheses. The option `-info=2` lists the predefined functions and constants.

Predefined functions		
Function	Description	Remarks
`abs()`	Absolute value, Modulus	
`acos()`	Arc cosine	
`asin()`	Arc sine	
`atan()`	Arc tangent	
`atan2()`	Angle function (arc tangent 2)	requires two arguments
`average()`	Average over all arguments	requires one or more arguments
`cos()`	Cosine	
`cosh()`	Hyperbolic cosine	
`cuberoot()`	Cube root (`cuberoot(a) = sgn(a)*abs(a)**1/3`)	
`exp()`	Natural exponential function (base e)	
`iff()`	Value selection	see Section 4.1.13
`log()`	Natural logarithm (base e)	
`log10()`	Common logarithm (base 10)	
`max()`	Maximum value	see Section 4.1.11
`maximum()`	Maximum value	see Section 4.1.11
`median()`	Median of all arguments	requires one or more arguments
`min()`	Minimum value	see Section 4.1.11
`minimum()`	Minimum value	see Section 4.1.11
`pow()`	Power (`pow(a,b) = a**b`)	see Section 4.1.12
`sgn()`	Sign (result: -1.0, 0.0 or 1.0)	
`sin()`	Sine	
`sinh()`	Hyperbolic sine	
`sqr()`	Square	
`sqrt()`	Square root	
`stdev()`	Standard deviation of all arguments	requires one or more arguments
`tan()`	Tangent	
`tanh()`	Hyperbolic tangent	

Predefined constants		
Constant	Description	Value
`e`	Euler's number	2.7182818284590451
`pi`	number pi	3.1415926535897931

Most of the predefined functions and constants are direct mappings of the NumPy functions with the same name. For details consult the NumPy documentation [9]. Most of the functions except `average()`, `median()`, `minimum()`, `min()`, `maximum()`, `max()`, `pow()` and `iff()` have one argument. All functions return one result.

`OMCE` allows to use special distribution functions in `<Equ>`, `<Proc>`, `<Function>` and `<Result>` elements. These functions are useful to model statistical effects without assigning a quantity name to it. This is particular useful in function definitions to create certain statistically independent properties every time the function is used. All parameters of the distribution functions must evaluate to constant values by either being a numerical constant or an arithmetic expression using only quantities which are defined as constants or by using the quantity access functions `val()` or `u()` (see below).

Predefined distribution functions		
Function	Description	Parameter
`Rectangle()`	Rectangular distribution	Mean value, half-width
`Normal()`	Normal distribution	Mean value, standard uncertainty
`Student()`	Student-ıt distribution	Mean value, scale factor, degrees of freedom
`Triangle()`	Triangular distribution	Mean value, half-width
`Trapez()`	Cosine distribution	Mean value, half-width, beta factor
`Ushape()`	U-shaped (arcus-sinus) distribution	Mean value, half-width
`Poisson()`	Poisson distribution	Mean value
`Ctrapez()`	Iso-curvilinear trapezoidal distr.	Mean value, half-width, beta factor
`Exponent()`	Exponential distribution	Mean value
`Gamma()`	Gamma distribution	Shape value, scale factor
`Laplace()`	Laplace distribution	Mean value, scale factor
`Lognormal()`	Lognormal distribution	Mean value, scale parameter
`Cauchy()`	Cauchy distribution	Mean value, scale factor

The `OMCE` defines two quantity access functions to access the mean value and the standard uncertainty of named input quantities. The result of a quantity access function is a constant. These functions can be used together with the distribution functions to model dependencies between the parameters of different distributions.

Predefined quantity access functions		
Function	Description	Parameter
`val()`	Value function to access the mean value of an input quantity	quantity name
`u()`	Uncertainty function to access the standard uncertainty of an input quantity	quantity name

`OMCE` supports evaluation models with multiple equations. The result of one equation may be used in the calculation of another equation. But no mathematical loops are allowed. It must be possible to evaluate the equation system in a linear execution order. The model will be checked for any mathematical loop before the simulation starts.

4.1.11 Function `minimum()` and `maximum()`

The functions `minimum()` (`min()`) and `maximum()` (`max()`) have one ore more parameter. The result of the function is the minimum or maximum of all the given values. These functions might produce a singular value in the PDF of the result if one of the parameter is a constant.

4.1.12 Function `pow()`

The function `pow()` can be alternatively used instead of the exponential operator `**`. The function has two parameters. The first is the basis and the second is the exponent.

4.1.13 Function `iff()`

The function `iff()` allows to switch between two values based on a specified condition. The function has three parameter. The first parameter is a logical expression. If the logical expression evaluates to `True` then the second parameter will be returned as a result. If the condition evaluates to `False` then the third parameter will be returned.

This function should be used with great care since it can produce serious discontinuities in the equation system. The use of this function might violate the basis for the application of GUM Supplement 1 [2] and the use is therefore not recommended. The function is provided for experimental purposes only.

4.1.14 Logical expressions

Logical expressions are used in constraints and as the first parameter of the `iif()`-function. They must evaluate to a boolean value (`True` or `False`). Usually logical expressions used in simulations compare data with limits or other data. The following comparison operators can be used:

Comparison operators			
Comparison	Operator	Alternative operator	Remarks
equal to	`==`	`.eq`	use with care in constraints
greater than	`>`	`.gt`	
less than	`<`	`.lt`	
greater than or equal to	`>=`	`.ge`	
less than or equal to	`<=`	`.le`	
not equal to	`!=`	`.ne`	

The characters < and > are used as special characters with XML-files and therefore cannot be entered directly in logical expressions. Use either the escape sequences `>` and `<` or the alternative dot operators instead.

Multiple comparisons can be combined with logical operators. The following operators are supported:

Logical operators	
Operator	Description
`and`	logical and
`or`	logical or
`not`	logical complement

Some examples for valid logical expressions:

```
X .ge 0
X_1 &lt; X_2
(X .ge 1) and (X .lt 2) and not(Y .lt 0)
```

The parenthesises are essential when combining comparisons.

4.1.15 Symbol names

Symbol names for quantities und equations are limited to the leters A...Z, a...z, the numbers 0...9 and the underscore character ("_"). Additionally, the @-character can be used, but it will be translated to a double underscore ("__") internally (e.g. A@A is the same as A__A). Therefore the @-character and the double underscore should not be used in the same model. The following words are reserved words and cannot be used as symbol names: and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield.

4.2 Binary data file formats

During the simulation a binary data file can be written (see option `-wb`). The `OMCE` supports two different binary formats.

4.2.1 `OMCE` binary format

The native `OMCE` binary format is a versatile binary format optimised for reading and writing speed. It has a block structure which corresponds to the processing block size of the data generation part of the simulation engine. The `OMCE` binary file has the following structure:

OMCE binary file format		
header	symbol table	binary data area
8 × 4 bytes	list of symbol names	runs × results × 8 bytes

The header is 32 bytes long and contains 6 × 32-bit integer values and 8 unused bytes. The header has the following structure:

OMCE binary header						
-1330463557	m results	n runs	block size b	data offset	flags	not used
32-bit int.	32-bit int.	32-bit int.	32-bit int.	32-bit int.	32 bits	2 × 4 bytes

The first integer in the header is a format identifier and contains the value -1330463557. The following fields describe the data structure in the binary data area and the number of bytes used for other purposes (offset to the data). The offset to the data should be a multiple of 8 bytes.

If bit 0 in the flag word is one then a symbol table is included in the binary file. If a symbol table is present, it will directly follow the header. The symbol table contains ASCII character strings with names terminated by a zero character (00). The end of the list is an empty string or two zero characters. The symbol table is padded with additional zero characters to fill a block with a size of a multiple of 8 bytes.

The binary data contains a number of data blocks. A data block contains double precision floating point values (64-bit IEEE 754-1985) for one result only. The size is determined by the block size (see option `-bs`). If more than one result is specified the first data block of the second result will follow the first data block of the first result and so on until all first data blocks for all results are written. The same scheme will be followed up to the last block of the last result. In case that the total number of runs is not a multiple of the block size b, then the last block will be smaller than the standard block size.

Binary data structure				
data block 0, result 0	...	data block 0 result m	...	data block k result m
b × 8 bytes	...	b × 8 bytes	...	b_{k+1} × 8 bytes

A binary file reader needs to read the header first to find out about the data structure. The total number of floating point values (64-bit) in the binary file is the product of results and runs $m \times n$ given in the header. The reader can determine the number of full data blocks k by dividing the number of runs by the block size given in the header n/b (integer division). In case the total number of values is not a multiple of the block size, then the last block is smaller and has a block size b_{k+1} which can be calculated from the number of runs modulo the block size $b_{k+1} = n \bmod b$ (modulo). After reading the header the reader should either read the symbol table if it is present or skip the number of bytes specified by the data offset in the header. The binary file contains information other than result values in this area. The value of the data offset might be zero. The unused integers in the header may be used later. The reader should not rely on them to have any specific value although they are set to zero by the `OMCE` so long they are not used.

A binary reader can distinguish between the `OMCE` binary format and the simple binary format (see next section)) by the value of the first entry in the header. The `OMCE` header contains a special negative number in this position while the number of runs n, the first integer in simple format, should always be positive.

4.2.2 Simple binary format

The simple binary format is an interleaving format like the `OMCE` format but with a fixed block size of 1 value per block. The simple binary file has the following structure:

Simple binary file format						
Header	result 0, value 0	result 1, value 0	...	result 0, value 1	...	result n, value m
2 × 4 bytes	8 bytes	8 bytes	...	8 bytes	...	8 bytes

The header contains 2 × 32-bit integer values and is followed by a series of alternating values for all results. The header has the following structure:

Simple binary header	
n runs	m results
32-bit integer	32-bit integer

A binary file reader for the simple format may read the header first to find out about the number of runs and the number of results in the data file. While reading the data file the reader might separate the values for the different results and form larger blocks of binary values per result to optimize the later data handling. Reading and writing in simple format is less efficient than in `OMCE` format. The conversion from internal arrays in `OMCE`to the simple interleaving format of one value per block and back consumes significant computational effort.

4.3 Text data file format

`OMCE` supports the generation of two different output files in comma separated value format. The ohd-file contains the histogram data and the var-file contains the link between the binary data file and the result quantity names given in the model description in the omc-file or xml-file and optional correlation information. An ohd-file is generated when `-a=1` is activated (default) and and a var-file is generated when `-wv` is set (deactivated by default). If a binary data file is written `-wb` <> 0 then option `-wv=1` should be set as well.

The content and the format of the text files (`.var` and `.ohd`) can be customized (see option `-ofd` and `-vfd`). The format of the different text output files is controlled by format definition files. The format definition files are text files themselves and they have a line based format. Every line in a format definition file is either a comment (starting with '#'), an option (starting with '-') or a column definition (having an equal sign). All lines which are not options and do not contain an equal sign are ignored.

This is the format definition for the `OMCE` var-file format:

```
#
# Native OMCE var-file format definition
#
-separator=","
-header=1
-extension=".var"
-quotechar="
SID=SID
ID=id
Symbol=name
Runs=runs
Binfile=filename
R[%s]=CORR
```

It defines a comma separated text file with the file name extension .var. The file has a header row and 5 or more columns named `SID`, `ID`, `Symbol`, `Runs`, `Binfile` and `R[0]` to `R[n-1]` for n results.

The following options should be used in the format definition files:

`-extension` This option defines the extension to the output file name. The extension should start with a point on the line. The value should be put into quotes.

`-header` This option controls whether or not the file contains a header line with the column headings. A value not equal 0 will generate a header line.

`-quotechar` This option controls the quoting character. No quotes should be used with this option.

`-separator` This option controls the character to separate the columns in the data file. Use the combination \t for the tab-character. The value should be put into quotes.

All lines which are neither a comment nor contain an option define a column in the data file if they contain an equal sign. The column name will be the string left of the equal sign and the content of the column is evaluated from the string right of the equal sign. The right side should contain a predefined symbol (see option `-info=3` for a list of implemented symbols and the source code for details).

The symbol `BINS`, `CORR` and `IRCORR` are used to specify the histogram data and correlation data. The string left of the equal sign must contain the sequence `%s` which will be replaced by an ident number or a name of an input quantity.

In case the string on the right side is quoted it will be taken literally and a fixed string is written in this column.

For array symbols (see option `-info=3`) two additional concepts are supported. Multiple data data can be included in the output file by specifying a range for the index in the array in the form `[<from>:<to>]` Both values can be omitted and the smallest and the largest index are used instead. If only a single number is given without the collon then only one array value is used.

The line `Blocksize=blocksize[0]` produces one column with the blocksize of the first result block and the line `Blocksize[%s]=blocksize[1:6]` produces 5 columns of blocksizes number 1 to 5 if the simulation has more than 5 result blocks.

Standard deviation and average are supported by using the `sd()` and the `avg()` function. The line `Blocksize=avg(blocksize)` includes a column with the average blocksize. (Note: In `OMCE` all blocks have the same size.)

5 Histogram viewer `OMCEview`

The `OMCE` package includes the simple histogram viewer `OMCEview` to show the result data as a histogram. `OMCEview` can be executed with the following command:

```
OMCEview <filename.ohd> [<options>]
```

The extension of the histogram data file must be given. `OMCEview` reads the histogram data file and shows histogram plots for the results as separate plots. The plots can be saved in various graphic formats. To run the source code version of the `OMCEview`, the python module matplotlib [10] should be installed.

5.1 Options

`OMCEview` supports the following options:

`-cfg` This option sets the name of a configuration file which contains default options in separate lines. The default value is `-cfg=OMCEview.cfg`.

`-fo` This option controls the file name of an output file (.pdf). By default the file is named after the ohd-file name by exchanging the extension.

`-h` This option controls the height of the diagrams in mm. The default height is 150 mm (`-h=150`).

`-it` Symbol names in axis labels and the title are typeset as italics if this option is activated `-it=1` (default).

`-mark` The option `-mark` specifies vertical markers in the diagram to mark for example the expectation value and the coverage interval. The value of this option must be a list of marker specifications enclosed in square brackets `[]` A marker specification consists of a comma separated list of four values enclosed in parenthesis defining the column name, the color, the line style and the line width of the marker. The default value is:

`-mark=[(Mean,b,dashed,1),(Low95,b,dashed,1),(High95,b,dashed,1)]`

The column names are referring to the names given in the ohd-file.

`-mm` A TEX like typesetting is used for symbols and units if this option is activated `-mm=1` (default).

`-max` This option specifies the limit for the maximum number of plots that will be generated based on the result data file (.ohd). The default is `-max=20`. If the number of results in the result data file exceeds the limit specified with this option, only the first results are plotted.

`-pdf` This option controls if the plots are printed to the screen (default) or printed to a multiple page pdf-file (`-pdf=1`). Screen plots can be saved to a file individually.

`-rm` Be default, a roman font is used to typeset text on the plots. Alternatively, a sanserif font can be used by setting this option to zero (`-rm=0`).

`-sht` This option controls the display of the diagram title. The diagram title is activated by default (`-sht=1`)

`-shx` This option controls the display of the x-axis label. The label is activated by default (`-shx=1`).

`-shy` This option controls the display of the y-axis label. The label is activated by default (`-shy=1`).

`-sid` This option specifies a range selection of simulation ident numbers (SID). The format of the option is `-sid=<min sid>:<max sid>` with `<min sid>` and `<max sid>` specify the lower and upper limit of simulation id's of the results which are plotted. The separator is a colon (":"). If any of the values are omitted then the minimum respectively the maximum SID in the file will be used. This option is only useful if the result data file contains data from different simulation runs.

`-sym` This option controls a symbol filter to select the result symbols which should be plotted. The filter is a comma separated list of symbol pattern. The wildcard characters `*` and `?` can be used. By default all symbols are selected (`-sym=*`). For example if the option `-sym=y_1,y_2,y_3` is given then only diagrams for y_1, y_2 and y_3 are plotted if the result symbols are present in the ohd-file.

`-yls` This option controls the font size in points of the diagram title. The default is `-xls=16.0`. In addition the following size names can be used: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`.

`-ttl` This option controls the content of the diagram title. By default (`-ttl=''`) the content of the column `Definition` is used. In case the column is not present in the ohd-file then the ohd-file name without extension is used instead. By setting this option, the user can overwrite the default by his own diagram title.

`-v` This option controls the verbose level between 0 for all messages and 10 for no messages. The default value is 2. For details see Section 3.1.

`-w` This option controls the width of the diagrams in mm. The default width is 200 mm (`-h=200`).

`-xl` This option controls the content of the x-axis label. By default (`-xl=''`) the content of the column `Symbol` is used as the label. By setting this option, the user can overwrite the default by his own label.

`-xls` This option controls the font size in points of the x-axis label. The default is `-xls=16.0`. In addition the following size names can be used: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`.

`-xts` This option controls the font size in points of the x-axis ticks. The default is `-xls=12.0`. In addition the following size names can be used: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`.

`-yl` This option controls the content of the y-axis label. By default (`-yl=''`) the content of the column ϕ(`Symbol`) is used as the label. By setting this option, the user can overwrite the default by his own label.

`-yls` This option controls the font size in points of the y-axis label. The default is `-xls=16.0`. In addition the following size names can be used: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`.

`-yts` This option controls the font size in points of the y-axis ticks. The default is `-xls=12.0`. In addition the following size names can be used: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`.

Note: By default the viewer uses the OpenType STIX fonts [11] to typeset title and labels. Warning messages will be generated if the fonts cannot be found on the system. The fonts are included in the OMCE distribution (`STIXv1.0.0\Fonts`). They should be copied to the system fonts directory.

6 Adaptive mode

The `OMCE` supports the automatic determination of the total number of Monte Carlo runs by the method described in the GUM Supplement 1 section 7.9 [2] (`-am=0`, default) or by alternate algorithms discuss in section 6.2. The adaptive mode is activated if the number of Monte Carlo blocks is set to zero (see option `-mcs=0`).

6.1 GUM Supplement 1 based algorithm

The GUM S1 based algorithm performs the simulation in two steps. In the beginning a number of initial runs n_{pr}, controlled by the option `-pr`, will be executed. After the initial runs the binning intervals are calculated and `OMCE` continues with performing additional runs.

After every additional run the tolerances T_s, T_h and T_l are calculated. Based on these tolerances `OMCE` decides when to terminate the simulations. The tolerances are calculated based on the number of significant digits n_{dig} controlled by the option `-sd` and the tolerance divisor d_a controlled by the option `-ad`.

For each data block i the average y_i, the standard deviation s_i, and the interval limits h_i and l_i are calculated. Based on the n_{r} data blocks simulated so far, the tolerances are calculated by

$$T_s = \frac{5 \times 10^{\mathrm{mag}(\bar{s}) - n_{\mathrm{dig}}}}{d_{\mathrm{a}}}$$

$$T_h = \frac{5 \times 10^{\mathrm{mag}(\bar{h} - \bar{y}) - n_{\mathrm{dig}}}}{d_{\mathrm{a}}}$$

$$T_l = \frac{5 \times 10^{\mathrm{mag}(\bar{y} - \bar{l}) - n_{\mathrm{dig}}}}{d_{\mathrm{a}}}$$

with

$$\bar{s} = \sqrt{\frac{1}{n_{\mathrm{r}}} \sum_{i=1}^{n_{\mathrm{r}}} s_i^2}, \quad \bar{h} = \frac{1}{n_{\mathrm{r}}} \sum_{i=1}^{n_{\mathrm{r}}} h_i, \quad \bar{y} = \frac{1}{n_{\mathrm{r}}} \sum_{i=1}^{n_{\mathrm{r}}} y_i, \quad \bar{l} = \frac{1}{n_{\mathrm{r}}} \sum_{i=1}^{n_{\mathrm{r}}} l_i.$$

The function $\mathrm{mag}(x)$ returns the order of magnitude of x.

During the additional simulation runs a multiple k of the standard deviation of the mean of the three parameters s_i, h_i and l_i are compared with their tolerances. The simulation terminates if all parameters are within their limits. The multiplier k is controlled by the option `-k`. With the total number of simulation runs n_{r} the termination criteria are

$$b_{\mathrm{MEAN}} = \left[\frac{k \times \mathrm{stddev}(y_1 \ldots y_n)}{\sqrt{n}} \leq T_s \right]$$

$$b_{\text{UNCERTAINTY}} = \left[\frac{k \times \text{stddev}(s_1 \ldots s_n)}{\sqrt{n}} \leq T_s\right]$$

$$b_{\text{LIMITS}} = \left[\frac{k \times \text{stddev}(h_1 \ldots h_n)}{\sqrt{n}} \leq T_h\right] \wedge \left[\frac{k \times \text{stddev}(l_1 \ldots l_n)}{\sqrt{n}} \leq T_l\right].$$

The function $\text{stddev}(x_1, \ldots, x_n)$ returns the experimental standard deviation of the valid data in the series $x_1 \ldots x_n$ (see Section 10.1).

Which criterion is actually in use is controlled by the option `-at`. The value of this option is a bit combination which enables the different criteria.

$$b_{\text{TERMINATION}} = (\neg\text{bit}(0) \vee b_{\text{MEAN}}) \wedge (\neg\text{bit}(1) \vee b_{\text{UNCERTAINTY}}) \wedge (\neg\text{bit}(2) \vee b_{\text{LIMITS}}).$$

The function bit() returns a logical value true if the bit in option `-at` with the number given as an argument is set and the value false otherwise. By default all termination criteria are used (`-at=7`).

6.2 Alternate algorithms

The `OMCE` supports alternate adaptive algorithms for the determination of the total number of Monte Carlo runs. The algorithms are chosen by the option `-am`. The standard GUM S1 algorithm is the default (`-am=0`).

The numerical accuracy of all algorithms is controlled by the same options. The number of significant digits n_{dig} are controlled by the option `-sd` and the tolerance divisor d_a is controlled by the option `-ad`. The probability level p for the test is controlled by the option `-af` and the block size n_b is controlled by the option `-bs`.

The algorithms selected with options `-am=1` and `-am=2` use a modified Stein-method [4]. After the initial n_{pr} simulation blocks controlled by option `-pr`, this methods estimates the total number of runs bases on a confidence interval.

The algorithms selected with options `-am=3` and `-am=4` are based on the fact that the numerical accuracy is in principle defined relative to the variance, which allows the exact knowledge about the number of runs needed for normally distributed results.

The accuracy definition in GUM S1 is defined as half of the least significant digit and therefore the relative accuracy is dependent on the actual digits of the uncertainty (between 0.5% for the digits "99" and 5% for digits "10") the number of runs needed to meet this specification is dependent on the digits of the uncertainty.

The algorithms selected with options `-am=1` and `-am=3` use the definition of the numerical accuracy as defined in GUM S1 while the algorithm selected with option `-am=2` and `-am=4` use always the highest accuracy (assuming the digits of the uncertainty are always "99").

The algorithm selected with option `-am=3` uses a simple scheme to adapt the number of runs according to the digits of the uncertainty.

At the end all algorithms perform a statistical test to ensure that the result meets the numerical accuracy even in cases when the results are not normally distributed.

The alternate algorithms support also the option `-at` to limit the checking of the statistical parameter to the mean (bit 0) or the variance (bit 1). Bit 2 has no function with these algorithms.

6.2.1 Sub-blocking

A sub-block mechanism is used to improve the statistical properties of the statistical evaluations. The result data blocks of size n_{b} are divided in sub-blocks of size n_{sb} (controlled by option `-sbs`) resulting in $n_{\text{s}} = n_{\text{r}} \times n_{\text{b}}/n_{\text{sb}}$ sub-blocks for n_{r} simulated blocks. The standard deviation of the values in sub-block number i is $s_i(y_{\text{sb}})$ and the mean value of the same block is $\bar{y}_{\text{sb}}(i)$. The degrees of freedom are improved by the ratio $n_{\text{b}}/n_{\text{sb}}$. The sub-blocking is implemented efficiently without any data copying by creating

a second view on the data. The sub-blocking is advantageous compare to a reduction of the block size because the performance of a Python based simulator and the accuracy of the evaluation of the coverage intervals are dependent on a large block size (≥ 10000).

6.2.2 Establishing the required statistical confidence level

As proposed in [4] `OMCE` uses a t-factor, based on the number of sub-blocks n_s, to achieve the numerical accuracy. As long as the sub-block mean values $\bar{y}_\mathrm{sb}(i)$ and the sub-block standard deviations $s_i(y_\mathrm{sb})$ can be considered as random samples of a distribution with an existing variance, the variance of the overall mean and the overall variance will be improved by a factor of n_s. Therefore it is possible to use the function $\mathrm{N_{sb}}(s, d)$ to calculate an estimate of the number of blocks needed to achieve the numerical accuracy based on the standard deviation s and the tolerance d.

$$\mathrm{N_{sb}}(s, d) = \max\left[\mathrm{int}\left(\frac{n_\mathrm{sb}}{n_\mathrm{b}} \times \frac{s^2 \times t_p(n_\mathrm{s}-1)^2}{d^2} + 0.5\right), 1\right]$$

The function $\max[n_1, \ldots, n_l]$ returns the maximum value of the values $n_1 \ldots n_l$ and the function $t_p(n_\mathrm{s}-1)$ returns the t-factor based on the probability p and the degrees of freedom $n_\mathrm{s} - 1$. In case n_s is large (> 100), the t-factor could be replaced by quantile function of a normal distribution.

6.2.3 Calculating the numerical tolerance

The numerical tolerances can be either established by using the GUM S1 definition that the tolerance is half of the least significant digit of the uncertainty or the tolerance can be defined as relative to the uncertainty of the result.

The tolerance for the mean value d_m based on the GUM S1 definition is

$$d_\mathrm{m} = 0.5 \times \frac{10^{\mathrm{mag}(u(y))+1} \times 10^{-n_\mathrm{dig}}}{d_a}$$

and the relative tolerance (in respect to the uncertainty) is dependent on the digits of the uncertainty (between 0.5% for the digits "99" and 5% for digits "10"). The function $\mathrm{mag}(x)$ returns the order of magnitude of x. This tolerance is used by the the algorithms selected with options `-am=1` and `-am=3`.

For the algorithms selected with options `-am=2` and `-am=4`, the relative tolerance (in respect to the uncertainty) is fixed (assuming the digits of the uncertainty are always "99").

$$d_\mathrm{m} = 0.5 \times \frac{u(y) \times 10^{-n_\mathrm{dig}}}{d_a}.$$

The tolerance for the variance d_v is for all algorithms

$$d_\mathrm{v} = 2 \times u(y) \times d_\mathrm{m}.$$

6.2.4 Predicting the number of simulation blocks

The algorithms selected with options `-am=3` and `-am=4` are based on the fact that if a mean value is calculated based on random samples from a distribution which has an existing variance then the ratio between the numerical accuracy of the mean value and the variance of the distribution is in principle predictable.

Based on the number of digits n_dig and the tolerance divisor d_a the relative tolerance factor m can be calculated.

$$m = 10^{n_\mathrm{dig}} \times d_a$$

Based on the Central Limit Theorem we can assume that the mean value is normally distributed if the number of simulations is large ($\gg 1000$) and therefore we use the quantiles of a normal distribution to calculate a confidence interval. The variance of the mean value is proportional to $1/n$ and we we can use this to predict the number of simulation runs n_p needed to establish the relative accuracy m (relative to the variance of the distribution). The predicted number of runs n_p divided by the block size n_b and rounded to the nearest integer greater than or equal to one is the number of simulation blocks n_0 needed to achieve the relative accuracy $1/m$ (in respect to the variance).

$$n_0 = \max\left[\mathrm{int}\left(\frac{m^2 \times \mathrm{norm}(1-(1-p)/2)^2 + n_\mathrm{b} - 1}{n_\mathrm{b}}\right), 1\right]$$

The function $\mathrm{norm}(p)$ calculates the quantile of the probability p of a normal distribution and the function $\mathrm{int}\,(x)$ returns the integer part of x.

The algorithm selected with option `-am=3` uses a multiple of n_0 to adapt the number of runs to the digits of the uncertainty.

The relative accuracy m in respect to the uncertainty is the correct numerical accuracy for the uncertainty digits "50" based on the GUM S1 definition. The correct numerical accuracy for the uncertainty digits "99" which is needed for the algorithm selected with option `-am=4`, results in a relative accuracy in respect to the uncertainty of $2 \times m$ and therefore in the number of simulation blocks n_1.

$$n_1 = \max\left[\mathrm{int}\left(\frac{4 \times m^2 \times \mathrm{norm}(1-(1-p)/2)^2 + n_\mathrm{b} - 1}{n_\mathrm{b}}\right), 1\right]$$

The value of n_0 is the number of simulation blocks needed to improve the variance of the mean value by m and n_1 is the number of simulation blocks needed to improve the variance of the mean value by $2 \times m$.

6.2.5 Algorithms based on the Stein-method

The algorithms selected with options `-am=1` and `-am=2` use a modified Stein-methods [4]. After the initial n_pr simulation blocks controlled by option `-pr`, this methods estimate the minimum number of runs needed bases on confidence intervals for the mean and the variance using the function $\mathrm{N_{sb}}$ defined in Section 6.2.2.

$$n_\mathrm{r} = \max\left[\mathrm{N_{sb}}(\mathrm{stddev}(\bar{y}_\mathrm{sb}(1), \ldots, \bar{y}_\mathrm{sb}(n_\mathrm{s})), d_\mathrm{m}), \mathrm{N_{sb}}(\mathrm{stddev}(s_1^2(y_\mathrm{sb}), \ldots, s_{n_\mathrm{s}}^2(y_\mathrm{sb})), d_\mathrm{v})\right].$$

The methods execute $n_\mathrm{r} - n_\mathrm{pr}$ additional runs and perform at the end the statistical test discussed in Section 6.2.7.

6.2.6 Algorithms based on a prediction of the needed runs

The algorithms selected with options `-am=3` and `-am=4` are based on a prediction of the runs needed (see Section 6.2.4).

If option `-am=3` or `-am=4` are selected then option `-pr` will be ignored and a number of $n_0 \times n_b$ simulations are run as initial runs.

After the initial n_0 runs the algorithm selected with option `-am=4` executes a fixed number of $n_1 - n_0$ runs as additional runs.

The algorithm selected with option `-am=3` uses the significant digits d_u of the uncertainty of the result $U(y)$ determined by the simulations so far to evaluate the need for additional runs.

$$d_u = \frac{u(y)}{10^{\mathrm{mag}(u(y))-1}}$$

The function $\mathrm{mag}(x)$ returns the order of magnitude of x.

The value of d_u is used to judge if additional runs are needed.

```
if (d_u < 10 + 10/m) or (d_u > 50 − 10/m):
    DoSimulation(n_0)
    if (d_u < 10 + 10/m) or (d_u > 70 − 10/m):
        DoSimulation(n_0)
        if (d_u < 10 + 10/m) or (d_u > 85 − 10/m):
            DoSimulation(n_0)
```

The function `DoSimulation(`n`)` executes n simulation blocks and m is the relative tolerance factor defined in Section 6.2.4. Note: The value of d_u is re-evaluated after the execution of `DoSimulation()`.

After the execution of the initial runs and the additional runs in total n_r simulation blocks have been executed. At the end both algorithms perform the statistical test (see next section).

6.2.7 Statistical end test

The improved adaptive algorithms selected by the option `-am` > 0 test the simulation results at the end to verify that the standard deviation of the mean and the standard deviation of the variance meet the tolerance specified by n_dig, d_a and p. The sub-blocking mechanism (controlled by option `-sbs`) is used to improve the statistical properties of this test (see Section 6.2.1).

The test verifies that the number of runs is greater than or equal to the square of the confidence interval based on a t-factor $t_p(n_\mathrm{s}-1)$ and the standard deviation divided by the square of the calculated tolerance.

The test uses the function $\mathrm{N_{sb}}(s,d)$ defined in Section 6.2.2 with the tolerances d_m and d_v discussed in Section 6.2.3.

The minimum number of runs n_{min} ensuring that the statistical properties for the mean value and the variance are met is

$$n_{min} = \max\left[\mathrm{N_{sb}}(\mathrm{stddev}(\bar{y}_\mathrm{sb}(1),\ldots,\bar{y}_\mathrm{sb}(n_\mathrm{s})),d_\mathrm{m}),\mathrm{N_{sb}}(\mathrm{stddev}(s_1^2(y_\mathrm{sb}),\ldots,s_{n_s}^2(y_\mathrm{sb})),d_\mathrm{v})\right].$$

The function $\mathrm{stddev}(x_1,\ldots,x_n)$ returns the experimental standard deviation of the valid data in the series $x_1 \ldots x_n$ (see Section 10.1) and the function $\max[n_1,\ldots,n_l]$ returns the maximum value of the values $n_1 \ldots n_l$.

An iterative test can be used at this stage to verify the statistical properties since the number of sub-blocks n_s is large (> 100).

```
while n_r < n_min:
    DoSimulation(1)
    n_r = n_r + 1
```

Note: The value of n_{min} is re-evaluated after every execution of `DoSimulation()`.

In case the sub-block size is large enough ($n_\mathrm{sb} \geq 200$) and the distribution of the result y is not extreme and the variance of the variance exists, one can assume a close to normal distribution for the mean values $\bar{y}_\mathrm{sb}(i)$ and the variances $s_i^2(y_\mathrm{sb})$ of the sub-blocks (Central Limit Theorem). In this case the standard deviations of the averages will be reduced by a factor of n_s and the total number of runs can be estimated or predicted by the algorithms very precisely. Then the statistical test at the end will find only in a small number of cases that the number of runs n_r is too small ($< 5\%$, one additional block). The need for a larger number of runs during the end test might be an indication that the Central Limit Theorem is not applicable for the specific simulation problem.

7 Running multiple simulations

The `OMCE` is a command line program that is designed to be used in batch mode. In case it is necessary to execute a Monte Carlo simulation several times based on the same underlying model, one can either use the batch processing of the operating system and call `OMCE` several times or one can use the section

<Simulations> in the omc-file to define multiple simulations. If some of the parameters need to be changed between the runs then they must be defined in the section <Parameters>. Using the batch processing of the operating system has the advantage that several simulations can run in parallel and that the same batch processing could be used for different models by changing only the omc-file name. Using the section <Simulations> has the advantage to combine everything in one input file and to generate one output file.

The following example demonstrates the use of the section Simulations to repeat the simulation three times and re-assigning the value of P to 1.0, 4.0 and 9.0.

```
<Simulations>
  <Simulation Name="First">
    <Parameter Name="P" Value="1.0"/>
  </Simulation>
  <Simulation Name="Second">
    <Parameter Name="P" Value="4.0"/>
  </Simulation>
  <Simulation Name="Third">
    <Parameter Name="P" Value="9.0"/>
  </Simulation>
</Simulations>
```

Alternatively, the same result can be achieved by using the element Loop.

```
<Simulations>
  <Loop Name="Loop" Enum="i" From="1" To="3">
    <Parameter Name="P" Value="i**2"/>
  </Loop>
</Simulations>
```

The value of a parameter can be evaluated using mathematical operators and predefined mathematical functions.

It should be noted that the parameter re-assignment in the section simulations override the parameter values given in the command line. So care should be taken if a omc-file with a section simulations is used together with batch processing.

8 Correlation coefficients between input quantities

The matrix based on the correlations coefficients given in the omc-file should be positive semi-definite. This is checked by Cholesky decomposition and eigenvalue decomposition. In cases when the matrix found to be not positive semi-definite the matrix is corrected by the chosen correction procedure (-mcp). The default procedure is to shift all eigenvalues which are smaller a given limit (-sev) to the limit and compose a new correlation matrix. A warning message together with the values of the modified correlation matrix is printed if a modification is necessary. The modification of the matrix is controlled by applying a norm [12] to the difference of the shifted and the non-shifted matrix. The norm must be smaller than a given limit (see option -e2) or the engine terminates with an error message.

9 Analysis of correlation

The OMCE can analyse the correlation in the results data. The analysis is controlled by the option -ac. The correlation is evaluated pair-wise by

$$r(q_i, q_j) = \frac{1}{(n-1)\, s(q_i)\, s(q_j)} \sum_{l=1}^{n} (q_{i,l} - q_i)\,(q_{j,l} - q_j)$$

with $q_{i,l}$ and $q_{j,l}$ being the data of the two results with the average q_i and q_j and the standard deviation $s(q_i)$ and $s(q_j)$. The correlation is evaluated for every block and the result correlation matrix is the average over all blocks. For m results $(m^2 - m)/2$ correlation coefficients are calculated. The correlation can be printed with the option `-lc=2` or the data can be written to the output files.

10 Constraints on the domain of the functions

The evaluation models given in the section `<Equations>` might be constrained in their domain. The use of the function `sqrt()` for example forms such constraint since the argument of the function cannot be negative. Any values violating any domain of the functions involved will be marked as not valid and the values will be ignore in further calculations. In addition, constraints specified in the section `<Constraints>` might restrict the domain and invalidate some values as well. As a consequence the number of valid values per block might be smaller than the number of trails. The number of runs in the statistical data file is always the number of valid values used to evaluate the result.

Since the statistical treatment is relying on a sufficiently large number of values per block, the ratio between valid values and trails is monitored. The simulation will be aborted as soon as the ratio drops below the limit defined by `-nl` (default 50 %). It is possible to reduce this limit, but it is advisable to increase the block-size (`-bs`) at the same time to guarantee that statistical treatment has still a good performance.

The correlation analysis can only be performed based on valid common data. Any invalid values in one quantity will automatically discard the values for al quantities from the data set. The ratio between the total number of values and the number of valid values is monitored. The simulation will be aborted as soon as the ratio drops below the limit defined by `-nc` (default 50 %).

The analysis of correlation can be disabled with the option `-ac=0` if the correlation information is not needed. It is also possible to reduce the limit with option `-nc` and increase the block-size sufficiently.

Non-valid data will reduce the performance of the simulation and therefore should be avoided if possible. The result will only be calculated from the valid data and all data marked as non-valid will not contribute to the result and therefore should not be calculated at all.

10.1 Mean value and uncertainty calculation

In case some of the data in a result data block i become not valid because of constraints, the effective block size of block i is reduced ($n_{\mathrm{eff}}(i) \leq n_{\mathrm{b}}$).

The total number of simulation results n in n_{r} simulation blocks is

$$n = \sum_{i=1}^{n_{\mathrm{r}}} n_{\mathrm{eff}}(i)$$

and the average effective blocksize $\bar{n}_{\mathrm{eff}}$ is

$$\bar{n}_{\mathrm{eff}} = \frac{1}{n_{\mathrm{r}}} \sum_{i=1}^{n_{\mathrm{r}}} n_{\mathrm{eff}}(i)$$

If $y[i]$ is the set of valid results for result data block i then the effective mean value $\bar{y}_{\mathrm{eff}}(i)$ of this set is

$$\bar{y}_{\mathrm{eff}}(i) = \frac{1}{n_{\mathrm{eff}}(i)} \sum_{j=1}^{n_{\mathrm{eff}}(i)} y[i]_j$$

and the effective variance $s^2_{\mathrm{eff}}(i)$ of the data in this set is

$$s^2_{\mathrm{eff}}(i) = \frac{1}{n_{\mathrm{eff}}(i) - 1} \sum_{j=1}^{n_{\mathrm{eff}}(i)} (y[i]_j - \bar{y}_{\mathrm{eff}}(i))^2.$$

The overall mean value of the result $\bar{y}$ is

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n_\mathrm{r}} n_\mathrm{eff}(i) \times \bar{y}_\mathrm{eff}(i)$$

and the variance $u^2(y)$ is

$$u^2(y) = \frac{1}{n-1} \sum_{i=1}^{n_\mathrm{r}} \sum_{j=1}^{n_\mathrm{eff}(i)} (y[i]_j - \bar{y})^2$$

which is equivalent of calculating

$$u^2(y) = \frac{1}{n-1} \left[\sum_{i=1}^{n_\mathrm{r}} (n_\mathrm{eff}(i) - 1) \times s^2_\mathrm{eff}(i) + \sum_{i=1}^{n_\mathrm{r}} n_\mathrm{eff}(i) \times (\bar{y}_\mathrm{eff}(i) - \bar{y})^2 \right].$$

The average of the effective variance $\bar{s}^2_\mathrm{eff}$ is

$$\bar{s}^2_\mathrm{eff} = \frac{1}{n} \sum_{i=1}^{n_\mathrm{r}} n_\mathrm{eff}(i) \times s^2_\mathrm{eff}(i).$$

The variance $s^2(\bar{y}_\mathrm{eff}())$ of the block mean values $\bar{y}_\mathrm{eff}(i)$ for $i = 1 \ldots n_\mathrm{r}$ is

$$s^2(\bar{y}_\mathrm{eff}()) = \frac{1}{\bar{n}_\mathrm{eff} \times (n_\mathrm{r} - 1)} \sum_{i=1}^{n_\mathrm{r}} n_\mathrm{eff}(i) \times (\bar{y}_\mathrm{eff}(i) - \bar{y})^2$$

and the variance of the effective variances of the sets is

$$s^2\left(s^2_\mathrm{eff}\right) = \frac{1}{\bar{n}_\mathrm{eff} \times (n_\mathrm{r} - 1)} \sum_{i=1}^{n_\mathrm{r}} n_\mathrm{eff}(i) \times (s_\mathrm{eff}(i) - \bar{s}_\mathrm{eff})^2.$$

The nominal variance $s^2_\mathrm{nom}(\bar{y}_\mathrm{eff}())$ of the block mean values $\bar{y}_\mathrm{eff}(i)$ for $i = 1 \ldots n_\mathrm{r}$ is

$$s^2_\mathrm{nom}(\bar{y}_\mathrm{eff}()) = \frac{\bar{n}_\mathrm{eff}}{n_\mathrm{b}} \times s^2(\bar{y}_\mathrm{eff}())$$

The nominal variance $s^2_\mathrm{nom}(y())$ is an estimate of the variance one would have got in case all blocks would have had the nominal size n_b.

11 Client server mode

The `OMCE` package supports a client server operation mode based on the RPyC [13] remote python call protocol. The server is called `OMCEserver` and it is implemented as an interface layer on top of `OMCE`. The interface layer provides the simulation service interface and handles the server command line options. A separate `OMCEclient` can be used as a client program. The `OMCEclient` is a lightweight client that deligates the simulation task to the server. The client only provides an interface for the server to the local file system and console I/O.

In a simple installation client and server may run on the same computer as a localhost installation (default). The protocol only needs an IP connection between client and server and the basic protocol is only designed to be used in a local area network behind firewalls where clients and servers can be trusted. By default the server does not use any encryption or authentication. For a more secure operation option `-TLS` or option `-VDB=` can be activated to enable authentication and encryption of the communication.

11.1 Running OMCE as a simulation server

The OMCEserver is started with the following command:

```
OMCEserver [options]
```

This will start OMCE in server mode, printing logging data to the console. The server will loop and wait for simulation jobs. All options except -CFG= can be set in the configuration file OMCEserver.cfg.

The server is controlled by the following options:

-AUTOREGISTER This option activates the automatic service registration of the OMCE simulation service at a registration server in the local network. This option is deactivated by default and it should be only activated if an RPyC registration server is available in the local network.

-CFG= This option sets the name of a configuration file which contains default options in separate lines. The default value is -CFG=OMCEserver.cfg.

-FORK By default OMCEserver uses a threaded server (see option -THREAD) which creates a new thread for every simulation request (connection). This is the default and the only option under Windows. Under Linux and other systems which support the fork() functionality this option can activate a forking server.

-IMPORTS= This option controls the directory for imports of binary data files. The client simulation can access this directory and its sub directory. The path given here is relative to the path of the OMCEserver executable. The default is -IMPORTS=imports.

-LOG= With this option the server activity can be logged to file. By default no log file is used. To log the server activity to a file use -LOG=log-file.log. By default all log files are saved in the sub directory logs under the directory with the OMCEserver executable. Under Linux one might want to use -LOG=/var/log/OMCE-server.log.

-NOAUTH This option deactivates any encryption and authentication protocal (default). To enable encryption and authentication use either option -TLS or option -VDB=.

-NOREGISTER This option deactivates any service registration (default). See option -AUTOREGISTER for details.

-PORT= This option controlls the port number to which the RPyC protocol is bound. The default is -PORT=4201. Note: The same port number must be used for client and server.

-Q , -Q- This option allows quiet (-Q) or non-quiet (-Q-) server operation. Note: It is sensible to activate a log-file with option -LOG= if option -Q is applied.

-THREAD This option activates a threaded server (default) which creates a new thread for every simulation request (connection). This is the only option under Windows. Under Linux and other systems which support the fork() functionality also option -FORK might be used.

-TLS This option activates a simple single user name with password authentication. The username and the password must follow this option seperated by spaces. Example: The option string "-TLS OMCE Gauss" sets the user name to OMCE and the password to Gauss.

-USER The option -USER activates the maintenance mode of the user verification database. All other options except the option -VDB= will be ignored. The option -USER must be followed by one of the commands given in the table below.

Command	Description
-USER add username password	Adds a new user to the database or changes the existing password
-USER addfile filename	Adds a list of users from a text file, with one user and password combination per line.
-USER del username	Deletes an existing user from the database.
-USER list	Shows a list of all existing users in the database.

The option `-VDB=vdb_file` must be given before the option `-USER`.

`-V=` This option controls the verbose level between 0 for all messages and 10 for no messages. The default value is 2. This option can be set to a lower level to see error tracebacks in the logs.

`-VDB=` This option activates a user verification database (dbm-clone). The filename of the database must be given after the equal sign. The path must be either an absolute file path or it is relative to the `OMCEserver` executable. The user verification database can be maintained with the `-USER` option.

Note: The server options all start with a double dash and use only capital letters. The switch option do not need an equal sign.

11.1.1 Limitations in client/server mode

The `OMCE` can perform the same simulations in client/server mode as it can perform in local mode. The complexity of the simulations is only limited by the memory and computing power of the server in client/server mode. A powerful server might be able to simulate a complex modell faster than the local machine. But performance will decrease dramatically if huge binary files of simulation data are transfered over the network. Therefore the option `-wb` is deactivated in server mode. Simulations must be run locally if a binary data file is needed.

The usage of the elements `<Discrete>` and `<Import>` in the OMC-file is supported in client/server mode, but might result in a poor performance. To improve performance the binary data should be copied to the server and the attribute `Syspath="1"` should be used to avoide the transfer of binary data over the network during simulation (see Section 4.1.9).

Note: For security reasons filenames starting with `"\"`, `"/"` or `".."` or which contain a colon (`":"`) are not permitted in client/server mode if the attribute `Syspath="1"` is used. Only binary files below the server directory `imports` can be accessed during the simulation.

11.2 Running `OMCEclient`

The `OMCEclient` is a lightweight client which provides a client interface to the simulation server. The `OMCEclient` is option-compatible with the `OMCE`. All options which can be used with `OMCE` (see section 3.1) can also be used with `OMCEclient`. Additional options control the server connection. Usually this options are set in the configuration file `OMCEclient.cfg`.

The connection between `OMCEclient` and the server is controlled by the following options:

`-cfg=` This option sets the name of a configuration file which contains default options in separate lines. The default value is `-cfg=OMCEclient.cfg`. This option cannot be used in a configuration file.

`-pass=` With this option a password can be set for the connection to the server. If either the password or the user name are set then a secured connection layer is used for the communication. By default no password is used.

`-port=` This option specifies the port which is used for the client/server connection. The defaul port is `-port=4201`.

`-server=` This option controls the servername or IP address of the `OMCE` server used for the client/server connection. The default server name is `-server=localhost`.

`-user=` This option specifies the user name which is used during authentication with the server. If either the this option or the password are set then a secured connection layer is used for the communication. By default the user name is empty.

If an OMCE server is running on the host and port specified in the configurarion file then the example from Section 3.2 can be executed with the following command.

```
OMCEclient S02.omc -pdf=1 -hc=400 -t=1
```

It will result in the same result file (S02.ohd) as the local invocation of OMCE.

11.3 Using client and server on different plattforms

The OMCE cleint/server mode can be used on different plattforms. For example clients running under Windows can use a server running under Linux and vice versa.

Note: The universal seperation character in a file path in the context of Python is the slash ("/") it can be used under Windows and Linux. Under Windows the backslash character ("\") can also be used, but this might create compatibility issues if the same OMC-file is used on different plattforms.

12 Running OMCE under Linux and other systems

The OMCE can be used under Linux and other systems if Python 2.5 or higher and all the packages are available on that system. To run OMCE recent SciPy and NumPy packages are needed which usually are not part of standard Linux distributions. Most of the packages can be successfully installed using easy_install but SciPy and NumPy should be installed using the source packages explicitly (avoiding easy_install).

12.1 Daemon wrapper

The OMCE includes an init.d compatible daemon wrapper OMCEd.py which provides OMCEserver functionality. As a basis OMCEd.py uses the same options as OMCEserver except -CFG= which is not supported. In addition some daemon specific options are supported.

OMCEd.py is started with the following command:

```
OMCEd [options]
```

This will start OMCEd.py as an independent daemon, printing logging data to a log-file if the option -LOG= is given. The daemon automatically starts a server which will loop and wait for simulation jobs. All options except -config can be set in a configuration file with the default name OMCEd_config.

In addition to the options for OMCEserver the following options are supported:

-config This option sets the name of a configuration file which contains default options in separate lines. The file name must follow as a separate argument without using an equal sign. The default value is -config OMCEd_config.

13 Used Python packages

The following packages are needed to run the OMCE with python.

Package	Description
Python	standard Python 2.7 environment [3]
NumPy	numerical extension [9]
SciPy	statistical functions [14]
Lxml	XML-file validation against the XSD-file [15]
Amara	XML-file processing [16]
Pyparsing	grammar definition for expressions [17]
RPyC	remote python call for cleint/server mode [13]
tlslite	TLS encryption and athentication for cleint/server mode [18]

For `OMCEview` the module matplotlib [19] is needed and py2exe [20] is needed to create EXE-versions of `OMCE` and `OMCEview`.

14 Testing and validation

`OMCE` makes use of the numerical library Numpy whenever possible. So the quality of the numerical calculations are based on this library. The library is widely used and tested and is considered reliable in the context of `OMCE` testing and validation.

The examples given in GUM Supplement 1 are included as examples and the results can be compared with the published results.

Output (`.ohd`) files for all examples can be found in `OMCE\examples\ref\`. They were generated with a seeded random generator (`-seed=1`). This is used as a basic regression test during the development and it should be used to verify the installation. The windows batch file `!run_all_OMCE.bat` runs all examples and compares the result with the reference results.

15 Performance

The `OMCE` is optimized for an efficient calculation of models with large equation systems. During the simulation a major part of the computational power of the system will be used. The memory consumption is dependent on the model and is about 1 MB per input quantity and result.

The binary data file can be rather large without being of much use after the simulation. By default the data file will be not created. Although the binary data file is written in parallel with the calculation, the time for writing the binary data file might contribute significantly to the overall execution time if the data calculation part is significantly faster than the data writing part. At the end the `OMCE` has to wait until all data blocks are written to the binary file before the simulation can be completed.

15.1 Random number generation

The `OMCE` uses the Mersenne Twister algorithm from the NumPy package. The Mersenne Twister algorithm has a large period ($2^{19937} - 1$), creates random floats to full 53-bit precision, and has been widely tested.

References

[1] Guide to the Expression of Uncertainty in Measurement, International Organization for Standardization, 1995

[2] Guide to the Expression of Uncertainty in Measurement Supplement 1: Numerical Methods for the Propagation of Distributions, JCGM Working Group of the Expression of Uncertainty in Measurement, 2006

[3] Python - Programming Language, open source software package, http://www.python.org

[4] G. Wübbeler, P. M. Harris, M. G. Cox, C. Elster: A two-stage procedure for determining the number of trials in the application of a Monte Carlo method for uncertainty evaluation, Metrologia 47(3): 317-324, 2010, doi:10.1088/0026-1394/47/3/023

[5] N. J. Higham: Computing the nearest correlation matrix - A problem from finance. IMA J. Numer. Anal., 22(3):329-343, 2002

[6] R. J. Hyndman, Y. Fan: Sample Quantiles in Statistical Packages, The American Statistician Vol. 50, No. 4 (1996), pp. 361-365, http://www.jstor.org/stable/2684934

[7] XML - Extensible Markup Language, flexible text format based on SGML (ISO 8879), http://www.w3.org/XML/

[8] R. N. Kacker, J. F. Lawrence: Rectangular distribution whose width is not exactly known: isocurvilinear trapezoidal distribution, Metrologia 46: 254ÂŰ260, 2009, doi:10.1088/0026-1394/46/3/012

[9] NumPy - Numerical Python extension, open source software package, http://numpy.scipy.org

[10] matplotlib - 2D plotting library, open source software package, http://matplotlib.sourceforge.net/

[11] STIX - OpenType fonts for Scientific and Technical Information eXchange, http://www.stixfonts.org/

[12] S. K. Mishra: Optimal solution of the nearest correlation matrix problem by minimization of the maximum norm, 2004, http://mpra.ub.uni-muenchen.de/1783

[13] RPyC - Remote Python Call for Python, open source software package, http://rpyc.wikidot.com

[14] SciPy - Scientific Tools for Python, open source software package, http://www.scipy.org

[15] Lxml - Python binding for the libxml2 and libxslt libraries, open source software package, http://codespeak.net/lxml

[16] Amara - Toolkit for processing XML in Python, open source software package, http://wiki.xml3k.org/Amara

[17] Pyparsing - Grammar definition language for Python, open source software package, http://pyparsing.wikispaces.com

[18] tlslite - SSL 3.0, TLS 1.0 and TLS 1.1 support for Python, open source software package, http://trevp.net/tlslite

[19] Matplotlib - 2D plotting library for Python, open source software package, http://matplotlib.sourceforge.net

[20] Py2exe - Windows executable generator for Python, open source software package, http://www.py2exe.org

A Program exit codes

Program exit codes	
Exit Code	Error message
001	Error: Option <opt> is not valid!
002	Error: File not found (<filename>)!
003	Error: It is not a file (<string>)!
004	Error: Format <format> is not supported!
005	Error: Option <option> should be a number (<value>)!
006	Error: Option out of range (<option>=<from>...<to>)!
007	Error: Bin-file format <format-id> not supported!
008	Error: Bin-file <filename> is corrupted (<details>)!
009	Error: No. of Runs in bin-file is zero!
010	Error: Mathematical loop in process list!
011	Error: Input quantity <name> is not defined!
012	Error: Duplicated quantity name (<name>)!
013	Error: Error in expression: <expression>!
014	Error: Quantity <name> in Result <name> is unknown!
015	Error: Model Parameter <parameter> not supported!
016	Error: Correlation matrix is not positive semi-definite!
017	Error: Quantity name unknown in correlation entry (<name>)!
018	Error: Correlations are not supported for non-Gaussian Quantities (<name>)!
019	Error: Correlation coefficient out of range r(<name>,<name>)=<value>!
020	Error: Matrix modification exceeds limit (-e2=<limit>)!
021	Error: Diagonal elements of the correlation matrix must be one [r(<name>,<name>)=<value>]!
022	Error: Quantity name <name> is a reserved symbol!
023	Error: Quantity <name>, parameter <param>, syntax error in <value>!
024	Error: Parameter of function <name>() is not a constant!
025	Error: Function <name> is used without parameter!
026	Error: Quantity unknown for val(<name>)!
027	Error: The uncertainty for a t-distribution with dof<3 is not defined in u(<name>)!
028	Error: Quantity unknown for u(<name>)!
029	Error: Grammar type not supported (<type-id>)!
030	Error: Function <name>() is used recursively in its definition!
031	Error: Quantity <name> is defined as parameter and as global in function <name>()!
032	Error: Global quantity <name> in function <name>() is not defined!
033	Error: Quantity <name> in function <name>() is not defined as parameter or global!
034	Error: Unknown command line parameter name <name>!
035	Error: Quantity <name>, parameter <param>, value <value> is not an integer!
036	Error: Quantity <name>, parameter <param>, parameter unknown in <value>!
037	Error: Quantity <name>, parameter <param>, %-character is not supported (<value>)!
038	Error: Quantity <name>, parameter <param>, syntax error in <value>!
039	Error: Quantity name <name> is a predefined function!
040	Error: Quantity name <name> is a predefined symbol!
041	Error: Quantity name <name> is a reserved name!
042	Error: Quantity name <name> is the name of a user defined function!
043	Error: Parameter name <name> is a predefined function!
044	Error: Parameter name <name> is a reserved name!
045	Error: Quantity <name> in <Discrete> or <Import> element is unknown in the bin-file <filename>!
046	Error: Index unknown!
047	Error: Result name <name> is used twice in result section!

Program exit codes	
Exit Code	Error message
048	Error: No result defined in source file!
049	Error: Unknown symbol <name> in parameter value <value>!
050	Error: Unknown symbol in parameter value <value>!
051	Error: Double assignment for r(<name>,<name>): <value> and <value>!
052	Error: A near correlation matrix could not be found in <runs> runs!
053	Error: Function <name> is used with <n> parameter!
054	Error: Format definition file <filename>: the line <line> cannot be decoded!
055	Error: Format definition header <header>, key <key> and function <func> cannot be interpretated!
056	Error: Less than <value> % of the simulated results are valid!
057	Error: The constraint for <name>: <expression> invalidates all data!
058	Error: Output File Definition not found (<filename>)!
059	Error: Constraint for <name>: <expression> cannot be evaluated!
060	Error: Unknown command line parameter argument <argument>!
061	Error: Parameter <param> in section <Simulation> is unknown!
062	Error: Loop increment (Step) must be non-zero!
063	Error: Parameter <param> in section <Loop> is unknown!
064	Error: Error in <element> name: <text>!
065	Error: quantile estimation method is unknown for -pq=<value>!
066	Error: The sub-block size -sbs=<value> must be a multiple of 10!
067	Error: The block size -bs=<value> must be a multiple of the sub-block size -sbs=<value>!
068	Error: Result data entry '<name>' unknown in ofd-file!
069	Error: Result data entry '<name>' unknown in vfd-file!
070	Error: The attribute mcsimulations is not set correctly (<value>)!
071	Error: The binary data format <number> is not supported!
072	Error: Internal, Matrix structure does not match!
254	Error: ConText I/O Error (<error message>)!
255	Error: <error message>!