JRP SIB59 Q-WAVE

A quantum standard for sampled electrical measurements

WP 4: Theoretical underpinning

XXX D4.4.1, Report:

Documentation to the toolbox for evaluation of digital/quantum generators errors

BETA version

CMI

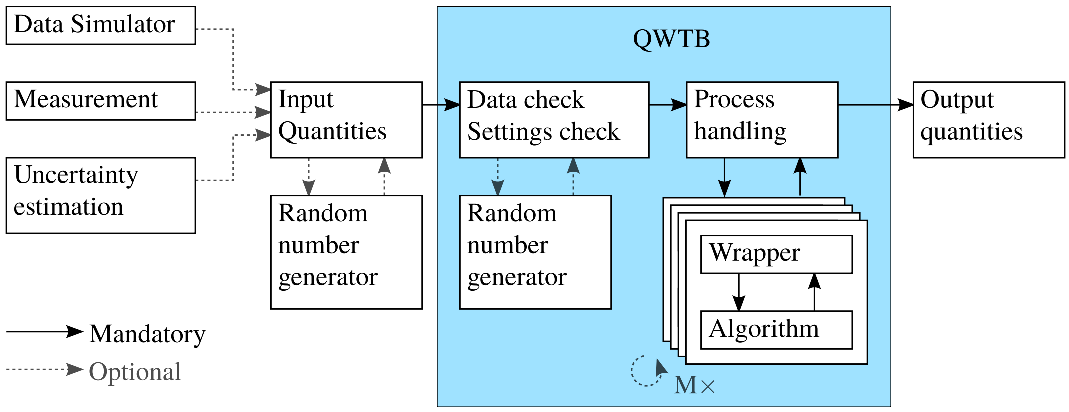
Contents

1 Introduction

Quantum Wave Tool Box (QWTB) is a toolbox for evaluation of measured data. QWTB consist of data processing algorithms from very different sources and unificating application interface. The toolbox gives the possibility to use different data processing algorithms with one set of data and removes the need to reformat data for every particular algorithm. Toolbox is extensible. The toolbox can variate input data and calculate uncertainties by means of Monte Carlo Method (MCM) [1].

2 Toolbox basic scheme

The basic scheme of the toolbox is following:



User have to prepare the data, either based on a real measurement or simulated, into a specified format. If needed, user can generate randomized data for selected quantities (e.g. with special probability density functions) and prepare for Monte Carlo uncertainty calculation. Next user calls toolbox to apply a selected algorithm on the data and review results. Toolbox will:

1. Check user data.
2. Check or generate calculation settings.
3. If required, quantities are randomized according uncertainties to prepare for MCM uncertainty calculation.
4. Data are handled to a wrapper. If needed, wrapper is run multiple times according MCM.
5. Output data are the result of the toolbox.

Another algorithm can be used immediately on the same data. User interface of the toolbox is represented by the function qwtb defined in the file `qwtb.m`.

3 Toolbox use

The toolbox is used in several modes according to a number and character of input arguments.

3.1 Get list of implemented algorithms

```
alginfo = qwtb()
```

With no input arguments, toolbox returns informations on all available algorithms. Result array alginfo contains structures for every algorithm found in the same directory as `qwtb.m`. Format of structures is defined in 5.

3.2 Application of an algorithm on the data

```
dataout = qwtb('algid', datain)
```

The algorithm is selected by first input argument algid. It is a string with designator of the algorithm, according structrue 5.

The second input argument is the user data. Data have to be formatted in a structure with fields named as quantities required by the algorithm (see 6).

The output variable is the structure with fields named as quantities.

In this case, standard calculation settings are used. If the user specifies calculation settings in structure according 7, it can be used as third input argument calcset:

```
dataout = qwtb('algid', datain, calcset)
```

For some calculation settings some fields of datain or calcset are generated automatically. To review automatically generated fields, user can get these structure in second and third output argument:

```
[dataout, datain, calcset] = qwtb('algid', datain)
[dataout, datain, calcset] = qwtb('algid', datain, calcset)
```

3.3 Running an example of algorithm use

Algorithm can have implemented an example of the use. This can be run by following syntax:

```
[] = qwtb('algid', 'example')
```

The algorithm is selected by first input argument algid. It is a string with designator of the algorithm, according structrue 5. The second argument is a string. Toolbox will run a script alg_example.m located in a algorithm directory.

After finish user can review input and output data or resulted figures if any.

3.4 Running a test of algorithm

Algorithm can have implemented a self test. This can be run by following syntax:

```
[] = qwtb('algid', 'test')
```

The algorithm is selected by first input argument algid. It is a string with designator of the algorithm, according structrue 5. The second argument is a string. Toolbox will run a script alg_test.m located in a algorithm directory.

Test should prepare data, run algorithm and check results. If implementation of algorithm behaves incorrectly, an error will occur.

3.5 Adding or removing algorithm path

Algorithms are stored in different directories, which are not in MATLAB/GNU Octave load path. To add directory with selected path to MATLAB/GNU Octave load path, following syntax is used:

```
[] = qwtb('algid', 'addpath')
```

To remove path, use:

```
[] = qwtb('algid', 'rempath')
```

Adding or removing path should be required only in special cases, such as debugging etc.

4 Algorithm directory structure implementation

Every algorithm is placed in a directory of following name:

`alg_X`

These directories have to be located in the directory containing the toolbox main script `qwtb.m`.

Every algorithm directory contains following files:

`X1`, `X2`, ... — Mandatory. One or more files with the algorithm itself.

`alg_info.m` — Mandatory. Description of the algorithm. See 4.1.

`alg_wrapper.m` — Mandatory. Wrapper of the algorithm. See 4.2.

`alg_test.m` — Recomended. Testing function. See 4.3.

`alg_example.m` — Recomended. Example script. See 4.4.

4.1 File `alg_info.m`

File contains a function with definition:

```
function alginfo = alg_info()
```

The output alginfo is a structure with informations about the algorithm. Structure is defined in 5.

File is mandatory. If file is missing in algorithm directory, QWTB will not recognize this algorithm as part of the toolbox.

4.2 File `alg_wrapper.m`

File contains a function with definition:

```
function dataout = alg_wrapper(datain, calcset)
```

The input datain is a structure with input data (see), calcset is a structure with definition of calculation settings (see 7).) and dataout is a structure containing output data (see).

The wrapper does following:

1. Formats input data structure datain into variables wuitable for algorithm.
2. Runs the algorithm.
3. Format results of the algorithm into data structure dataout.

File is mandatory. If file is missing in algorithm directory, QWTB will not recognize this algorithm as part of the toolbox.

4.3 File alg_test.m

File contains a function with following definition:

```
function [] = alg_test(calcset)
```

Test should generate sample data, run algorithm and check results by a function assert. QWTB will provide a standard calculation settings structure calcset (see 7).

This file is not mandatory, however is recommended.

4.4 File alg_example.m

Example contains a script showing a basic use of the algorithm. The format of the file should conform to the publishing markup defined in Matlab documentation. See matlab help on keyword *Publishing markup*). The QWTB runs this script in base context, thus all variables defined in the example script will be accessible to the user.

To create a documentation of the QWTB, function publish is applied to the example script and resulting file is attached to the documentation file.

5 Algorithm informations structure

Structure defines properties and possibilities of the algorithm. All fields are mandatory but . fullpath .

.id — Designator of the algorithm.

.name — Name of the algorithm.

.desc — Basic description.

.citation — Reference.

.remarks — Any remark.

.requires — Required quantities.

.reqdesc — Short description of required quantities.

.returns — Output quantities.

.retdesc — Short description of output quantities.

.providesGUF — Algorithm/wrapper calculates GUF uncertainty.

.providesMCM — Algorithm/wrapper calculates MCM uncertainty.

.fullpath — Full path to the algorithm. Automatically generated by the toolbox.

5.1 .id

String. Designator of the algorithm. It is unique identifier, no two algorithms can have same id.

5.2 .longname

String. Full name of the algorithm.

5.3 .desc

String. Basic description of the algorithm.

5.4 .citation

String. A reference to the paper, book or other literature with full description of the algorithm.

5.5 .remarks

String. Remarks, license or others related to the algorithm.

5.6 .requires

Cell array of strings. Names of quantities required by the algorithm.

5.7 .reqdesc

Cell array of strings. Short description of quantities required by the algorithm.

5.8 .returns

Cell array of strings. Names of quantities returned by the algorithm.

5.9 .retdesc

Cell array of strings. Short description of quantities returned by the algorithm.

5.10 .providesGUF

Boolean. If nonzero, the wrapper or the algorithm calculates uncertainty by means of GUM Uncertainty Framework.

5.11 .providesMCM

Boolean. If nonzero, the wrapper or the algorithm calculates uncertainty by means of Monte Carlo Method.

5.12 .fullpath

String. Full path to the algorithm. This field is automatically generated by QWTB.

6 Quantity structure

Every quantity is a structure with following fields:

.v — Value.

.u — Uncertainty.

.d — Degree of freedom.

.c — Correlation.

.r — Randomized uncertainty.

6.1 .v

Value of the quantity. Can be a scalar, *row* vector or matrix. More dimensions are not supported.

6.2 .u

Standard uncertainty of the quantity. Dimensions are the same as of the value field.

6.3 .d

Degrees of freedom the uncertainty according GUM Uncertainty Framework. Dimensions are the same as of the value field.

This field is automatically generated by the toolbox if missing, required and calcset.dof.gen is set to nonzero. The value will be set to 50.

6.4 .c

Correlation matrix for quantity. 2DO XXX.

This field can be automatically generated by the toolbox if missing, required and calcset.cor.gen is set to nonzero. The value will be set to 0.

6.5 .r

Randomized uncertainties according Monte Carlo method. In the case of scalar quantity it is *column* vector of length equal to calcset.mcm.repeats. For a vector quantity it is a matrix with number of columns equal to length of value of the quantity and number of rows equal to calcset.mcm.repeats. For a matrix quantity it is a matrix with three dimensions, first two equal to the dimensions of value quantity, third dimension equal to calcset.mcm.repeats.

This field is required if Monte Carlo uncertainty calculation is required. In this case it can be automatically generated by the toolbox if missing and calcset.mcm.randomize is set to boolean. The pdf will be normal, sigma will be equal to the standard uncertainty of the quantity.

6.6 Quantity structure examples

Example of scalar quantity of mean value 1, standard uncertainty 0.1, degrees of freedom 9, correlation has no sense for scalar quantity, and radnomized matrix has number of elements equal to calcset.mcm.randomize.

$$\begin{array}{ll}
\text{.v:} & \begin{pmatrix}1\end{pmatrix} \\
\text{.u:} & \begin{pmatrix}0.1\end{pmatrix} \\
\text{.d:} & \begin{pmatrix}9\end{pmatrix} \\
\text{.c:} & \begin{pmatrix}0\end{pmatrix} \\
\text{.r:} & \begin{pmatrix}1.02076 \\ 1.22555 \\ \vdots \\ 0.89727\end{pmatrix}
\end{array}$$

Example of vector quantity with i elements, M is equal to calcset.mcm.randomize (only symbolic representation):

$$
\begin{array}{ll}
\text{.v:} & (v_1, v_2, \ldots, v_i) \\
\text{.u:} & (u_1, u_2, \ldots, u_i) \\
\text{.d:} & (d_1, d_2, \ldots, d_i) \\
\text{.c:} & \begin{pmatrix} c_{11} & \ldots & c_{1i} \\ \vdots & \ddots & \vdots \\ c_{i1} & \ldots & c_{ii} \end{pmatrix} \\
\text{.r:} & \begin{pmatrix} r_{11} & \ldots & r_{1i} \\ \vdots & \ddots & \vdots \\ r_{M1} & \ldots & r_{Mi} \end{pmatrix}
\end{array}
$$

Example of matrix quantity with i times j elements, M is equal to calcset .mcm. randomize (only symbolic representation):

$$
\begin{array}{ll}
\text{.v:} & \begin{pmatrix} v_{11} & \ldots & v_{1j} \\ \vdots & \ddots & \vdots \\ v_{i1} & \ldots & v_{ij} \end{pmatrix} \\
\text{.u:} & \begin{pmatrix} v_{11} & \ldots & u_{1j} \\ \vdots & \ddots & \vdots \\ u_{i1} & \ldots & u_{ij} \end{pmatrix} \\
\text{.d:} & \begin{pmatrix} d_{11} & \ldots & d_{1j} \\ \vdots & \ddots & \vdots \\ d_{i1} & \ldots & d_{ij} \end{pmatrix} \\
\text{.c:} & (XXX???) \\
\text{.r:} & \begin{pmatrix} r_{111} & \ldots & r_{1j1} \\ \vdots & \ddots & \vdots \\ r_{i11} & \ldots & r_{ij1} \end{pmatrix} \\
 & \vdots \\
 & \begin{pmatrix} r_{11M} & \ldots & r_{1jM} \\ \vdots & \ddots & \vdots \\ r_{i1M} & \ldots & r_{ijM} \end{pmatrix}
\end{array}
$$

7 Calculation settings structure

Structure defines calculation methods.

.strict — (0) If zero, other fields generated automatically.

.verbose — (1) Display various informations.

.unc — ('none') How uncertainty is calculated ('none', 'guf', 'mcm').

.cor.req — (0) Correlation matrix is required for all input quantities.

.cor.gen — (1) Zero correlation matrix is generated automatically if missing.

.dof.req — (1) Degrees of freedom are required for all input quantities.

.dof.gen — (1) Degree of freedom are generated automatically if missing with value 50.

.mcm.repeats — (100) Number of Monte Carlo iterations.

.mcm.verbose — (1) Display various informations concerning Monte Carlo method.

.mcm.method — ('singlecore') Parallelization method ('multicore', 'multistation').

.mcm.procno — (1) Number of processors to use.

.mcm.tmpdir — ('.') Directory for temporary data.

.mcm.randomize — (1) Randomized uncertainties are generated automatically if missing.

7.1 .strict

Boolean, default value 0. If set to zero, all other fields of the structure are generated automatically and set to a default value.

7.2 .verbose

Boolean, default value 1. If set to non-zero value, various messages are displayed during calculation, such as used uncertainty calculation method, automatic generation of matrices etc.

7.3 .unc

String, default value ''. Determines uncertainty calculation method. Only three values are possible:

'' — Uncertainty is not calculated.

'guf' — Uncertainty is calculated by GUM Uncertainty Framework [2].

'mcm' — Uncertainty is calculated by Monte Carlo Method [1].

See chapter XXX for uncertainty calculation details.

7.4 .cor

Structure sets handling of correlation matrices of quantities. Structure has two fields:

.req — Boolean, default value 0. If non-zero, correlation matrices are required for all quantities.

.gen — Boolean, default value 1. If non-zero, correlation matrices will be generated automatically if missing in quantity.

Automatically generated correlation matrices has all elements of zero value.

7.5 .dof

Structure sets handling of degrees of freedom of quantities. Structure has two fields:

.req — Boolean, default value 0. If non-zero, degrees of freedom are required for all quantities.

.gen — Boolean, default value 1. If non-zero, degree of freedom will be generated automatically if missing in quantity.

Automatically generated degree of freedom has value 50.

7.6 .mcm

Structure sets handling of Monte Carlo calculation of uncertainties. Structure has following fields:

.repeats — Positive non-zero integer, default value 100. Number of iterations of Monte Carlo method.

.verbose — Boolean, default value 1. If set to non-zero value, various messages are displayed during calculation of Monte Carlo method such as used parallelization method, number of calculated iterations etc.

.method — String, default value 'singlecore'. Parallelization method used for Monte Carlo method calculation. Only three values are possible:

'singlecore' — No parallelization, all is calculated on one CPU core.

'multicore' — Calculation is divided into cores of one computer.

'multistation' — Calculation is distributed on several computers.

Not all methods are possible to use on all computers. 'singlecore' is always possible to use. 'multicore' use parfor in Matlab or parcellfun in GNU Octave. 'multistation' use

.procno — Zero or positive integer, default value 0. Number of CPU cores exploitable by the parallelization method. If set to zero, all available CPU cores will be used. If desktop computer is used, it is good practice to set to number of CPU cores minus one, so the computer can be used by other tasks also.

.tmpdir — String, default value '.' (current directory). Temporary directory for storing temporary data needed for some parallelization methods.

.randomize — Boolean, default value 1. If non-zero, randomized uncertainties will be generated automatically if missing, but only if uncertainty calculation method is set to 'mcm' (Monte Carlo) to prevent large memory usage.

8 Bilbiography

[1] JCGM, *Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method*, JCGM, Ed. Bureau International des Poids et Measures, 2008.

[2] ——, *Evaluation of measurement data - Guide to the expression of uncertainty in measurement*, JCGM, Ed. Bureau International des Poids et Measures, 1995, ISBN: 92-67-10188-9.

9 Quick reference

Toolbox use:

```
alginfo = qwtb()
dataout = qwtb('algid', datain)
[dataout, datain, calcset] = qwtb('algid', datain)
dataout = qwtb('algid', datain, calcset)
[dataout, datain, calcset] = qwtb('algid', datain, calcset)
[] = qwtb('algid', 'example')
[] = qwtb('algid', 'test')
[] = qwtb('algid', 'addpath')
[] = qwtb('algid', 'rempath')
```

Algorithm informations structure:

.id — Designator of the algorithm.
.name — Name of the algorithm.
.desc — Basic description.
.citation — Reference.
.remarks — Any remark.
.requires — Required quantities.
.reqdesc — Description of required quantities.
.returns — Output quantities.
.retdesc — Description of output quantities.
.providesGUF — Algorithm/wrapper calculates GUF uncertainty.
.providesMCM — Algorithm/wrapper calculates MCM uncertainty.
.fullpath — Full path to the algorithm. Automatically generated by the toolbox.

Quantity structure:

.v — Value.
.u — Uncertainty.
.d — Degree of freedom.
.c — Correlation.
.r — Randomized uncertainty.

Calculation settings structure:

.strict — (0) If zero, other fields generated automatically.
.verbose — (1) Display various informations.
.unc — ('none') How uncertainty is calculated ('none', 'guf', 'mcm').
.cor.req — (0) Correlation matrix is required for all input quantities.
.cor.gen — (1) Zero correlation matrix is generated automatically if missing.
.dof.req — (1) Degrees of freedom are required for all input quantities.
.dof.gen — (1) Degree of freedom are generated automatically if missing with value 50.
.mcm.repeats — (100) Number of Monte Carlo iterations.
.mcm.verbose — (1) Display various informations concerning Monte Carlo method.
.mcm.method — ('singlecore') Parallelization method ('multicore', 'multistation').
.mcm.procno — (1) Number of processors to use.
.mcm.tmpdir — ('.') Directory for temporary data.
.mcm.randomize — (1) Randomized uncertainties are generated automatically if missing.

testG

Table of Contents

Example for algorithm testG. Algorithm is usefull only for testing QWTB toolbox. It calculates maximal and minimal value of the record. GUF is calculated by wrapper.

See also `qwtb`

Generate sample data

Two quantities are prepared: `x` and `y`.

```
x = []; y = [];
x.v = [1:20];
y.v = [1:14 13:-1:8];
```

All uncertainties are set to 1.

```
x.u = x.v.*0 + 1;
y.u = y.v.*0 + 1;
```

Set degrees of freedom.

```
x.d = x.v.*0 + 60;
y.d = y.v.*0 + 9;
```

Quantities are put into data input structure `DI`.

```
DI = [];
DI.x = x;
DI.y = y;
```

Create calculation settings `CS` and set uncertainty calculation method to GUM uncertainty framework.

```
CS = [];
CS.unc = 'guf';
```

Call algorithm

Use QWTB to apply algorithm `testG` to data `DI` with calculation settings `CS`.

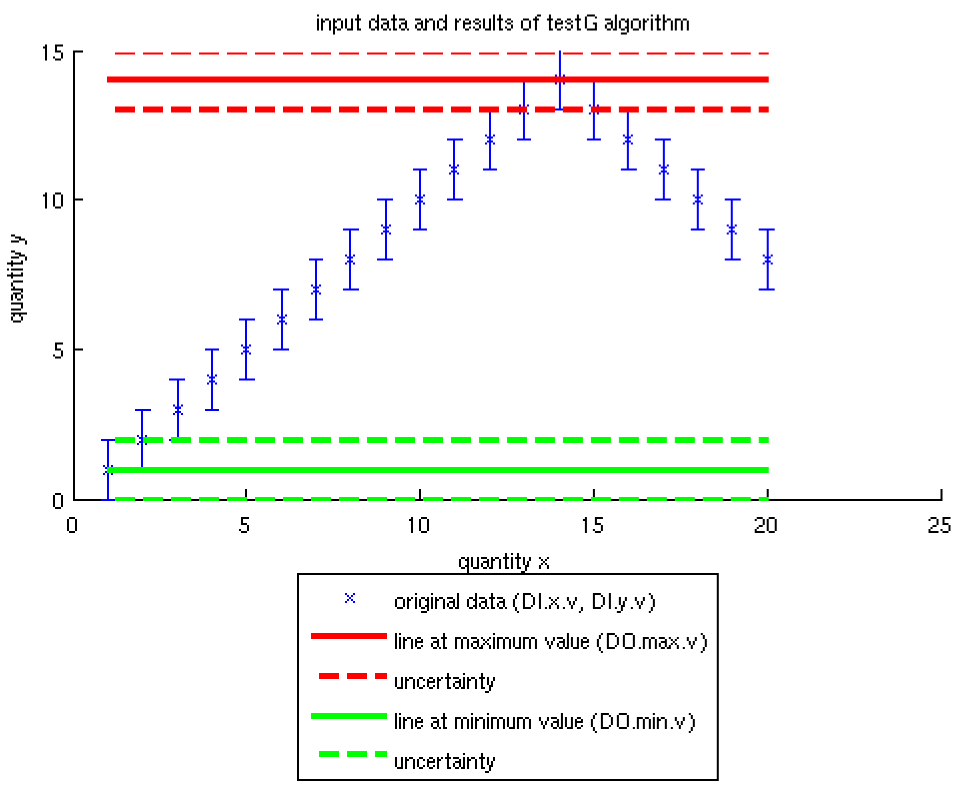
```
DO = qwtb('testG', DI, CS);
```

```
QWTB: default correlation matrix generated for quantity `x`
QWTB: default correlation matrix generated for quantity `y`
QWTB: uncertainty calculation by means of wrapper or algorithm
```

Plot results

Plot input data and calculated maximal and minimal values as a red and green lines with uncertainties represented by dashed lines.

```
figure
hold on
errorbar(DI.x.v, DI.y.v, DI.y.u, 'xb')
plot([DI.x.v(1) DI.x.v(end)], [DO.max.v DO.max.v], '-r', 'linewidth', 3)
plot([DI.x.v(1) DI.x.v(end)], [DO.max.v - DO.max.u DO.max.v - DO.max.u], '--r', 'l
plot([DI.x.v(1) DI.x.v(end)], [DO.min.v DO.min.v], '-g', 'linewidth', 3)
plot([DI.x.v(1) DI.x.v(end)], [DO.min.v - DO.min.u DO.min.v - DO.min.u], '--g', 'l
plot([DI.x.v(1) DI.x.v(end)], [DO.max.v + DO.max.u DO.max.v + DO.max.u], '--r', 'l
plot([DI.x.v(1) DI.x.v(end)], [DO.min.v + DO.min.u DO.min.v + DO.min.u], '--g', 'l
legend('original data (DI.x.v, DI.y.v)', 'line at maximum value (DO.max.v)', 'unce
xlabel('quantity x')
ylabel('quantity y')
title('input data and results of testG algorithm')
hold off
```



Published with MATLAB® R2013a

testGM

Table of Contents

Example for algorithm testGM. Algorithm is usefull only for testing QWTB toolbox. It calculates maximal and minimal value of the record. GUF/MCM is calculated by wrapper.

See also `qwtb`

Generate sample data

Two quantities are prepared: `x` and `y`.

```
x = []; y = [];
x.v = [1:20];
y.v = [1:14 13:-1:8];
```

All uncertainties are set to 1.

```
x.u = x.v.*0 + 1;
y.u = y.v.*0 + 1;
```

Set degrees of freedom.

```
x.d = x.v.*0 + 60;
y.d = y.v.*0 + 9;
```

Quantities are put into data input structure `DI`.

```
DI = [];
DI.x = x;
DI.y = y;
```

Create calculation settings `CS` and set uncertainty calculation method to Monte Carlo method. Allow randomization of uncertainties by the QWTB toolbox.

```
CSMCM = [];
CSMCM.unc = 'mcm';
CSMCM.mcm.randomize = 1;
```

Create calculation settings and set uncertainty calculation method to GUM uncertainty framework.

```
CSGUF = [];
CSGUF.unc = 'guf';
```

Call algorithm

Use QWTB to apply algorithm `testGM` to data `DI` with calculation settings `CSGUF`.

```
DOGUF = qwtb('testGM', DI, CSGUF);
```

```
QWTB: default correlation matrix generated for quantity `x`
QWTB: default correlation matrix generated for quantity `y`
QWTB: uncertainty calculation by means of wrapper or algorithm
```

Use QWTB to apply algorithm `testGM` to data `DI` with calculation settings `CSMCM`.

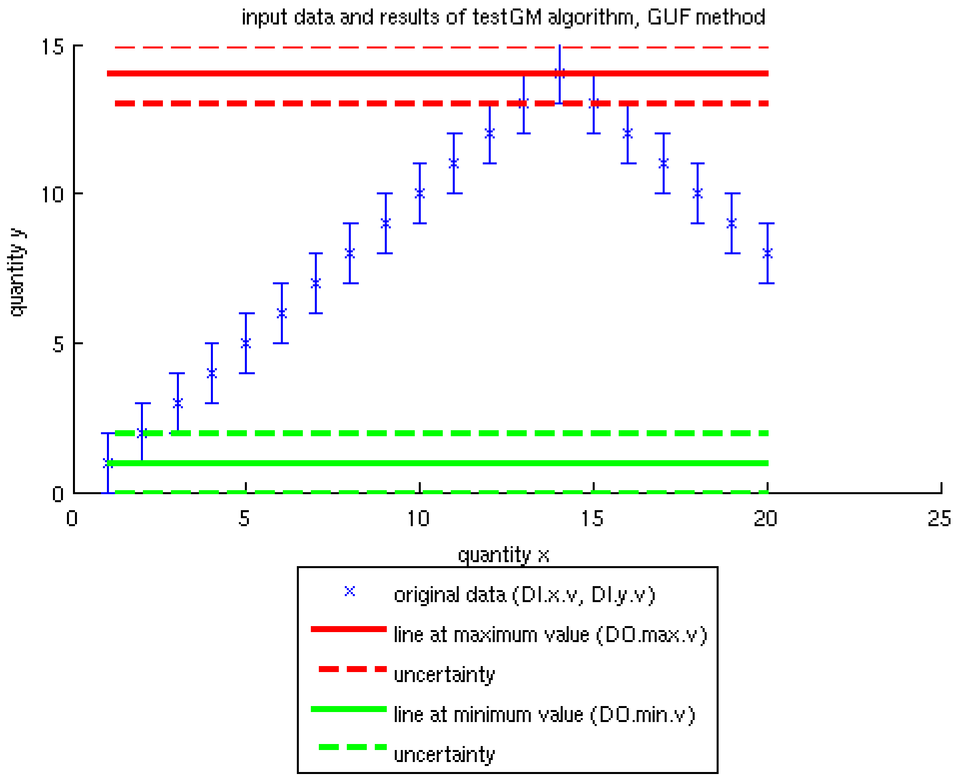
```
DOMCM = qwtb('testGM', DI, CSMCM);
```

```
QWTB: default correlation matrix generated for quantity `x`
QWTB: quantity x was randomized by QWTB
QWTB: default correlation matrix generated for quantity `y`
QWTB: quantity y was randomized by QWTB
QWTB: uncertainty calculation by means of wrapper or algorithm
```

Plot results

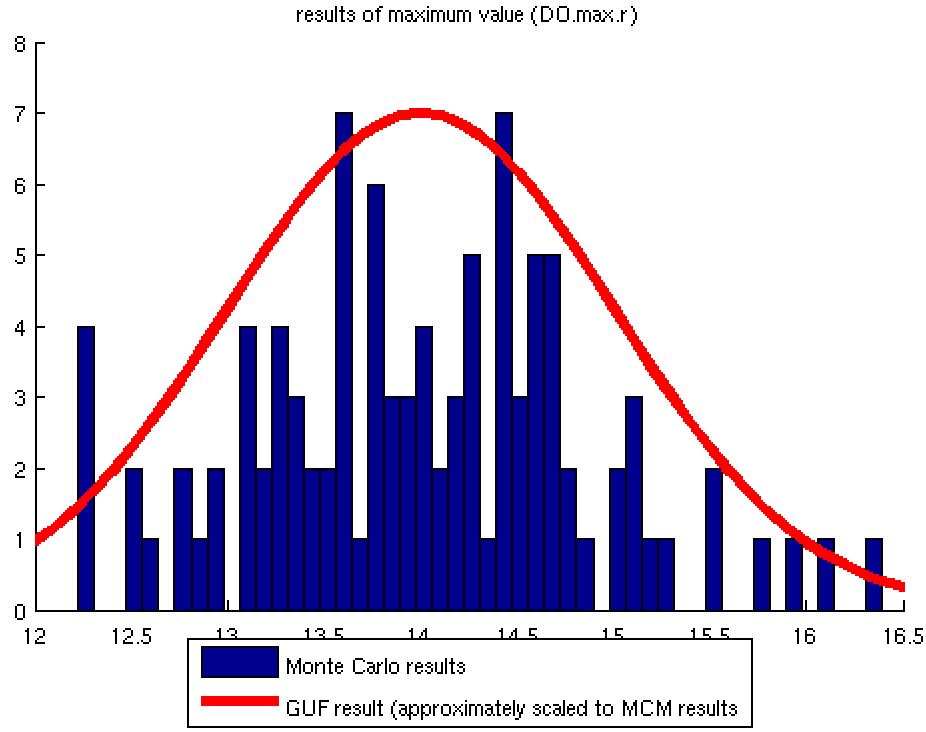
Plot input data and calculated maximal and minimal values as a red and green lines with uncertainties represented by dashed lines.

```
figure
hold on
errorbar(DI.x.v, DI.y.v, DI.y.u, 'xb')
plot([DI.x.v(1) DI.x.v(end)], [DOGUF.max.v DOGUF.max.v], '-r', 'linewidth', 3)
plot([DI.x.v(1) DI.x.v(end)], [DOGUF.max.v - DOGUF.max.u DOGUF.max.v - DOGUF.max.u
plot([DI.x.v(1) DI.x.v(end)], [DOGUF.min.v DOGUF.min.v], '-g', 'linewidth', 3)
plot([DI.x.v(1) DI.x.v(end)], [DOGUF.min.v - DOGUF.min.u DOGUF.min.v - DOGUF.min.u
plot([DI.x.v(1) DI.x.v(end)], [DOGUF.max.v + DOGUF.max.u DOGUF.max.v + DOGUF.max.u
plot([DI.x.v(1) DI.x.v(end)], [DOGUF.min.v + DOGUF.min.u DOGUF.min.v + DOGUF.min.u
legend('original data (DI.x.v, DI.y.v)', 'line at maximum value (DO.max.v)', 'unce
xlabel('quantity x')
ylabel('quantity y')
title('input data and results of testGM algorithm, GUF method')
hold off
```



Plot histogram of calculated maximal value, i.e. probability density function simulated by Monte Carlo method and overlay by result of GUF method (approximately scaled to MCM result).

```
figure
hold on
hist(DOMCM.max.r, 50)
a = axis;
x = [a(1):0.1:a(2)];
pdf = normpdf(x, DOGUF.max.v, DOGUF.max.u);
plot(x, a(4)/max(pdf).*pdf, '-r', 'linewidth', 4);
title('results of maximum value (DO.max.r)')
legend('Monte Carlo results', 'GUF result (approximately scaled to MCM results', '
hold off
```



Published with MATLAB® R2013a

testM

Table of Contents

Example for algorithm testM. Algorithm is usefull only for testing QWTB toolbox. It calculates maximal and minimal value of the record. MCM is calculated by wrapper.

See also `qwtb`

Generate sample data

Two quantities are prepared: `x` and `y`.

```
x = []; y = [];
x.v = [1:20];
y.v = [1:14 13:-1:8];
```

All uncertainties are set to 1.

```
x.u = x.v.*0 + 1;
y.u = y.v.*0 + 1;
```

Quantities are put into data input structure `DI`.

```
DI = [];
DI.x = x;
DI.y = y;
```

Create calculation settings `CS` and set uncertainty calculation method to Monte Carlo method. Allow randomization of uncertainties by the QWTB toolbox.

```
CS = [];
CS.unc = 'mcm';
CS.mcm.randomize = 1;
```

Call algorithm

Use QWTB to apply algorithm `testM` to data `DI` with calculation settings `CS`.

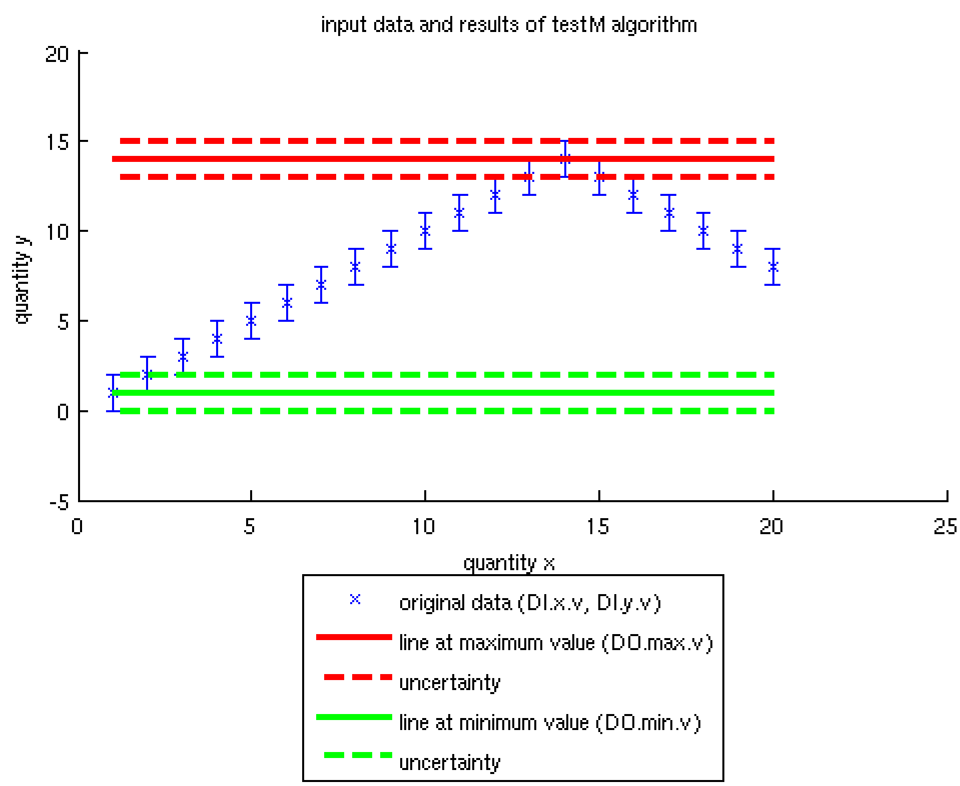
```
DO = qwtb('testM', DI, CS);
```

```
QWTB: default correlation matrix generated for quantity `x`
QWTB: quantity x was randomized by QWTB
QWTB: default correlation matrix generated for quantity `y`
QWTB: quantity y was randomized by QWTB
QWTB: uncertainty calculation by means of wrapper or algorithm
```

Plot results

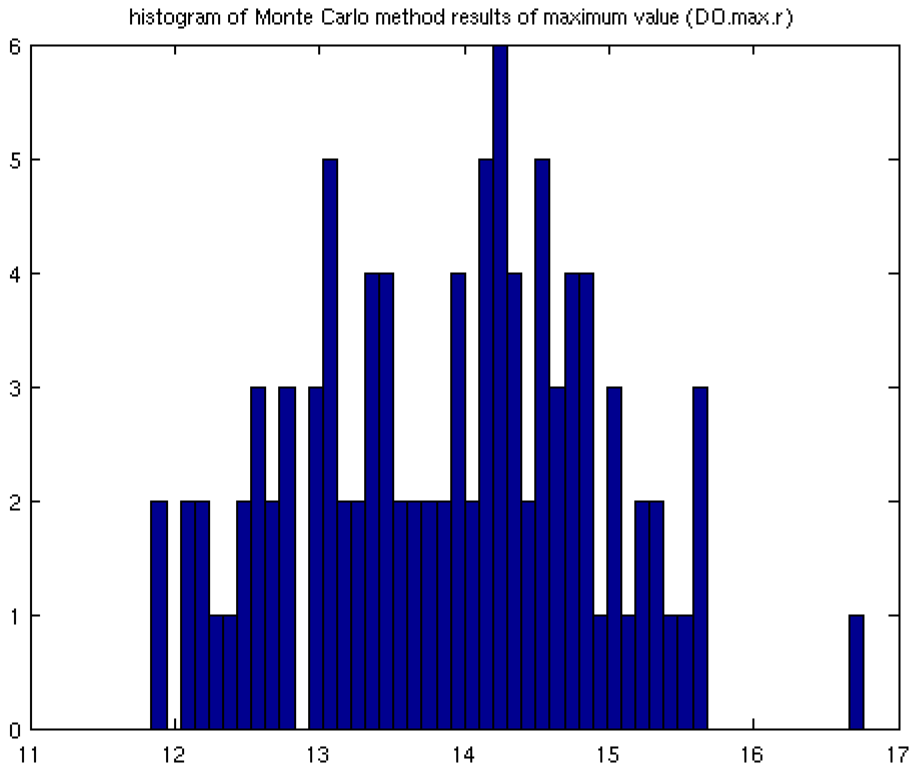
Plot input data and calculated maximal and minimal values as a red and green lines with uncertainties represented by dashed lines.

```
figure
hold on
errorbar(DI.x.v, DI.y.v, DI.y.u, 'xb')
plot([DI.x.v(1) DI.x.v(end)], [DO.max.v DO.max.v], '-r', 'linewidth', 3)
plot([DI.x.v(1) DI.x.v(end)], [DO.max.v - DO.max.u DO.max.v - DO.max.u], '--r', 'l
plot([DI.x.v(1) DI.x.v(end)], [DO.min.v DO.min.v], '-g', 'linewidth', 3)
plot([DI.x.v(1) DI.x.v(end)], [DO.min.v - DO.min.u DO.min.v - DO.min.u], '--g', 'l
plot([DI.x.v(1) DI.x.v(end)], [DO.max.v + DO.max.u DO.max.v + DO.max.u], '--r', 'l
plot([DI.x.v(1) DI.x.v(end)], [DO.min.v + DO.min.u DO.min.v + DO.min.u], '--g', 'l
legend('original data (DI.x.v, DI.y.v)', 'line at maximum value (DO.max.v)', 'unce
xlabel('quantity x')
ylabel('quantity y')
title('input data and results of testM algorithm')
hold off
```



Plot histogram of calculated maximal value, i.e. probability density function simulated by Monte Carlo method.

```
figure
hist(DO.max.r, 50)
title('histogram of Monte Carlo method results of maximum value (DO.max.r)')
```



Published with MATLAB® R2013a