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`.

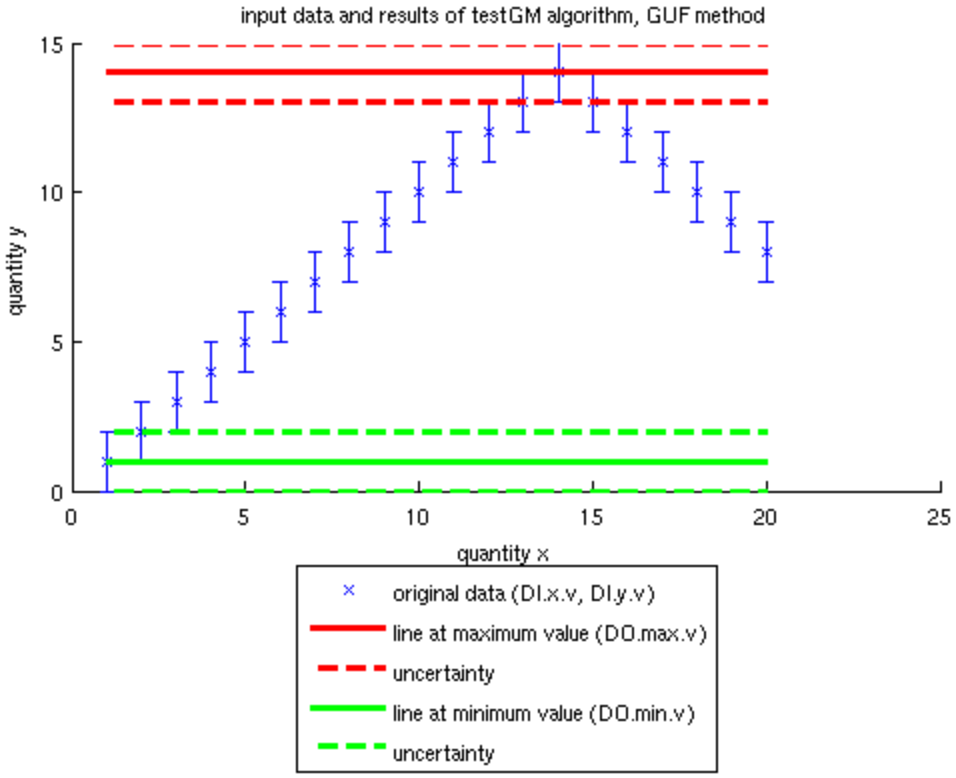
```
DOMCM = qwtb('testGM', DI, CSMCM);
```

```
        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

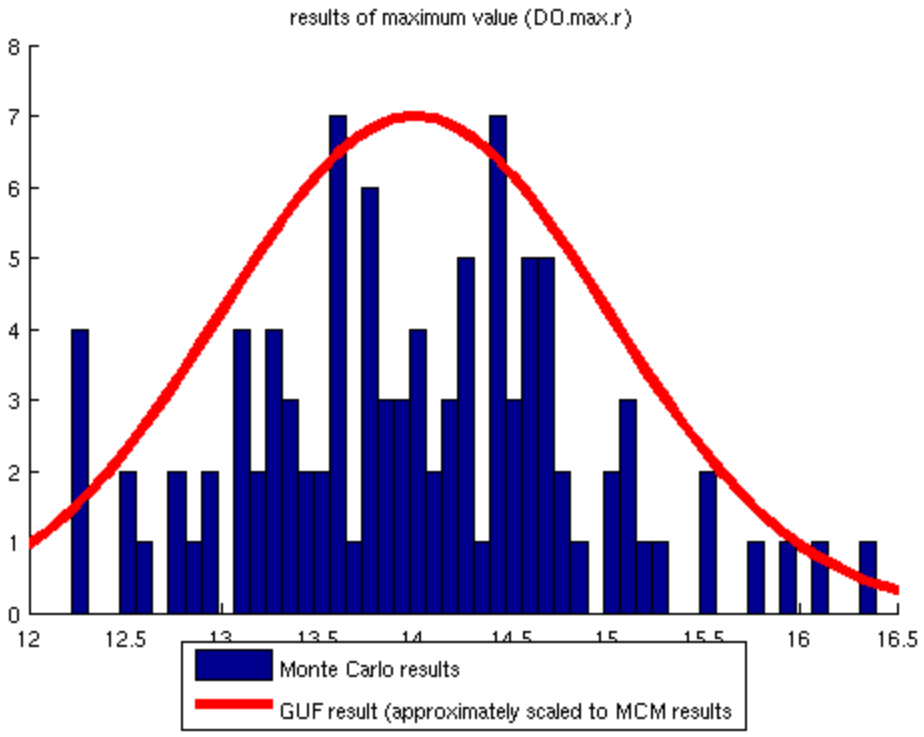
Plot input data and calculated maximal and minimal values as a red and green lines with uncertainties represented by dashed lines.

```
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