



Wollensack Michael 18.11.2009

---

# Metas.UncLib.Matlab V1.0

## User Reference

---

This document is a quick reference sheet. For practical demonstrations and more details refer to the tutorial and the examples that are provided with the installation of the software.

The *Metas.UncLib.Matlab* library is an extension to MATLAB, which supports creation of uncertainty objects and subsequent calculation with them as well as storage of the results. It has been developed with MATLAB V7.5.0 and it requires the C# library *Metas.UncLib* in the background. The classes 'LinProp', 'DistProp' and 'MCProp' wrap *Metas.UncLib* to MATLAB over the COM Interface.

- 'LinProp' supports linear uncertainty propagation. This class is fully functional.
- 'DistProp' supports higher order uncertainty propagation, i.e. higher order terms of the Taylor expansion of the measurement equation are taken into account. This class currently only supports Gaussian input distributions and assumes Gaussian result distributions. In the future it will be possible to specify non-Gaussian inputs and to use higher moments of the results to calculate more realistic coverage intervals.
- 'MCProp' supports Monte Carlo propagation but its implementation is still preliminary and its use is not recommended yet.

## 1. Global uncertainty settings

### 1.1 Set function handle

<code>unc = @LinProp</code>	Set function handle <code>unc</code> to linear uncertainty propagation.
<code>unc = @DistProp</code>	Set function handle <code>unc</code> to higher order uncertainty propagation.
<code>unc = @MCPProp</code>	Set function handle <code>unc</code> to Monte Carlo uncertainty propagation.

### 1.2 Additional global settings

<code>g = DistPropGlobal;</code> <code>g.MaxLevel</code>	Set the higher order uncertainty propagation maximum level. Default value: 1 (1 corresponds to LinProp)
<code>g = MCPPropGlobal;</code> <code>g.n</code>	Set the Monte Carlo uncertainty propagation sample size. Default value: 100'000

## 2. Create an uncertainty object

Square brackets indicate vector or matrix.

<code>unc(value)</code>	Create uncertain number or array without uncertainties
<code>unc(value, standard_unc, (idof))</code>	Creates a new real uncertain number with value, standard uncertainty and inverse degrees of freedom (optional).
<code>unc(value, [covariance], (idof))</code>	Creates a new complex uncertain number. Covariance size: 2x2
<code>unc([value], [covariance], (idof))</code>	Creates a new real uncertain array. N: numel(value) Covariance size: NxN
<code>unc([value], [covariance], (idof))</code>	Creates a new complex uncertain array. N: numel(value) Covariance size: 2Nx2N
<code>unc(value, [sys_inputs], [sys_sensitivities], 'system')</code>	Create uncertain number by setting sensitivities with respect to uncertain inputs. <sup>1</sup>

## 3. Calculations with uncertainty objects

Use MATLAB call `methods(y)` on uncertainty object `y` to obtain a full list of supported methods.

### 3.1 Math functions

<code>x + y</code>	<code>sqrt(x)</code>	<code>sin(x)</code>	<code>sinh(x)</code>	<code>real(x)</code>
<code>x - y</code>	<code>exp(x)</code>	<code>cos(x)</code>	<code>cosh(x)</code>	<code>imag(x)</code>
<code>x.*y</code>	<code>log(x)</code>	<code>tan(x)</code>	<code>tanh(x)</code>	<code>abs(x)</code>
<code>x./y</code>	<code>log10(x)</code>	<code>asin(x)</code>	<code>asinh(x)</code>	<code>angle(x)</code>
<code>x.^y</code>	<code>log(x, y)</code>	<code>acos(x)</code>	<code>acosh(x)</code>	<code>conj(x)</code>
	<code>sign(x)</code>	<code>atan(x)</code>	<code>atanh(x)</code>	<code>atan2(x, y)</code>

## 3.2 Linear Algebra

<code>M1*M2</code>	Matrix multiplication of matrix M1 and M2
<code>lu(M)</code>	LU decomposition of matrix M
<code>det(M)</code>	Determinate of matrix M
<code>inv(M)</code>	Matrix inverse of M
<code>A\Y</code>	Solve linear equation system: $A * X = Y$
<code>A\Y</code>	Least square solve over determined equation system.
<code>lscov(A, Y, W)</code>	Weighted least square solve over determined equation system.
<code>fft(v)</code>	Fast Fourier transformation
<code>ifft(v)</code>	Inverse Fast Fourier transformation

## 4. Get properties of an uncertainty object

<code>get_value(y)</code>	Returns the expected value.
<code>get_fcn_value(y)</code>	Returns the function value.
<code>get_stdunc(y)</code>	Computes the standard uncertainty.
<code>get_idof(y)</code>	Computes the inverse degrees of freedom. <sup>1</sup>
<code>1./get_idof(y)</code>	Computes the degrees of freedom. <sup>1</sup>
<code>get_coverage_interval(y, p)</code>	Computes the coverage interval.
<code>get_moment(y, n)</code>	Computes the n <sup>th</sup> central moment.
<code>get_jacobi(y)</code>	Returns the sensitivities to the virtual base inputs (with value 0 and uncertainty 1). <sup>1</sup>
<code>get_jacobi2(y, x)</code>	Computes the sensitivities of y to the intermediate results x. <sup>1</sup>
<code>get_unc_component(y, x)</code>	Computes the uncertainty components of y with respect to x. <sup>1</sup>
<code>get_correlation([y1 y2 ...])</code>	Computes the correlation matrix
<code>get_covariance([y1 y2 ...])</code>	Computes the covariance matrix

## 5. Storage functions

### 5.1 Store a computed uncertainty object

<code>binary_file(y, filepath)</code>	Binary serializes uncertainty object y to file.
<code>xml_file(y, filepath)</code>	Xml serializes uncertainty object y to file.
<code>xml_string(y)</code>	Xml serializes uncertainty object y to string.

### 5.2 Reload a stored uncertainty object

<code>unc(filepath, 'binary_file')</code>	Reloads uncertainty object from binary file.
<code>unc(filepath, 'xml_file')</code>	Reloads uncertainty object from xml file.
<code>unc(xml_string)</code>	Reloads uncertainty object from xml string.

<sup>1</sup> 'LinProp' uncertainty objects only