METAS UncLib C# - User Reference V2.4.3

Michael Wollensack

December 2020

Contents

1 Introduction

This document is a quick reference sheet. For more details refer to the METAS UncLib help with the installation of the software.

The METAS UncLib C# library supports creation of uncertainty objects and subsequent calculation with them as well as storage of the results. It's able to handle complex-valued and multivariate quantities. It has been developed with Microsoft Visual Studio 2013 and it requires the .NET Frame-work V4.5. There are three namespaces for uncertainty propagation: `LinProp`, `DistProp` and `MCProp`.

LinProp supports linear uncertainty propagation $\mathbf{V}_{out} = \mathbf{J}\mathbf{V}_{in}\mathbf{J}'$.

DistProp supports higher order uncertainty propagation, i.e. higher order terms of the Taylor expansion of the measurement equation are taken into account.[1]

MCProp supports Monte Carlo propagation.[1]

2 Global uncertainty settings

2.1 Using Metas.UncLib

```
using Metas.UncLib.Core;
using Metas.UncLib.LinProp;
```
Use the linear uncertainty propagation.

```
using Metas.UncLib.Core;
using Metas.UncLib.DistProp;
```
Use the higher order uncertainty propagation.

```
using Metas.UncLib.Core;
using Metas.UncLib.MCProp;
```
Use the Monte Carlo uncertainty propagation.

2.2 Additional global settings

`DistProp.Misc.Global.MaxLevel` Set the higher order uncertainty propagation maximum level. Default value: 1 (1 corresponds to `LinProp`)

`MCProp.Misc.Global.n` Set the Monte Carlo uncertainty propagation sample size. Default value: 100000

[1] preliminary implementation

3 Create an uncertainty object

Square brackets indicate vector or matrix.

`UncNumber x = new UncNumber(value);` Creates a new uncertain number without uncertainties.

`UncNumber x = new UncNumber(value, standard_unc, (idof));` Creates a new real uncertain number with value, standard uncertainty and inverse degrees of freedom (optional).

`UncNumber x = Unc.RealUncNumber(value, standard_unc, (idof));` Creates a new real uncertain number with value, standard uncertainty and inverse degrees of freedom (optional).

`Complex<UncNumber> x = Unc.ComplexUncNumber(value, [covariance], (idof));` Creates a new complex uncertain number. Covariance size: $2x2$

`UncNumber[] x = Unc.RealUncArray([value], [covariance], (idof));` Creates a new real uncertain array. Covariance size: NxN

`Complex<UncNumber>[] x = Unc.ComplexUncArray([value], [covariance], (idof));` Creates a new complex uncertain array. Covariance size: $2Nx2N$

`UncNumber x = new UncNumber(); x.Init(value, [sys_inputs], [sys_sensitivities]);` Create uncertain number by setting sensitivities with respect to uncertain inputs.[2]

4 Calculations with uncertainty objects

4.1 Math functions

- `x + y`
- `x * y`
- `x - y`
- `x / y`
- `-x`
- `Math.Sign(x)`
- `Math.Sqrt(x)`
- `Math.Sin(x)`
- `Math.Sinh(x)`
- `Math.Real(x)`
- `Math.Exp(x)`
- `Math.Cos(x)`
- `Math.Cosh(x)`
- `Math.Imag(x)`
- `Math.Log(x)`
- `Math.Tan(x)`
- `Math.Tanh(x)`
- `Math.Abs(x)`
- `Math.Log10(x)`
- `Math.Asin(x)`
- `Math.Asinh(x)`
- `Math.Angle(x)`
- `Math.Log(x, y)`
- `Math.Acos(x)`
- `Math.Acosh(x)`
- `Math.Conj(x)`
- `Math.Pow(x, y)`
- `Math.Atan(x)`
- `Math.Atanh(x)`
- `Math.Atan2(x, y)`

[2] `LinProp` uncertainty objects only

4.2 Linear algebra

`LinAlg.Dot(M1, M2)` Matrix multiplication of matrix $\mathbf{M_1}$ and $\mathbf{M_2}$

`LinAlg.Lu(M)` LU decomposition of matrix $\mathbf{M}$

`LinAlg.Det(M)` Determinate of matrix $\mathbf{M}$

`LinAlg.Inv(M)` Matrix inverse of $\mathbf{M}$

`LinAlg.Solve(A, Y)` Solve linear equation system: $\mathbf{Ax} = \mathbf{y}$

`LinAlg.LstSqrSolve(A, Y)` Least square solve over determined equation system

`LinAlg.WeightedLstSqrSolve(A, Y, W)` Weighted least square solve over determined equation system

`UncLinAlg.NonLinearEig(A)` Non-linear eigenvalue problem[2]: $\mathbf{A_0V} + \mathbf{A_1VD} + \mathbf{A_2VD}^2 + \ldots + \mathbf{A_{(n-1)}VD}^{(n-1)} = 0$

4.3 Numerical routines

`NumLib.PolyFit(x, y, n)` Fit polynom to data

`NumLib.PolyVal(p, x)` Evaluate polynom

`NumLib.Interpolation(x, y, n, xx)` Interpolation

`NumLib.Interpolation2(x, y, n, xx)` Interpolation with linear uncertainty propagation

`NumLib.SplineInterpolation(x, y, xx, boundaries)` Spline interpolation

`NumLib.SplineInterpolation2(x, y, xx, boundaries)` Spline interpolation with linear uncertainty propagation

`NumLib.Integrate(x, y, n)` Integrate

`NumLib.SplineIntegrate(x, y, n)` Spline integrate

`NumLib.Fft(v)` Fast Fourier transformation

`NumLib.Ifft(v)` Inverse Fast Fourier transformation

[2] `LinProp` uncertainty objects only

5 Get properties of an uncertainty object

`Unc.GetValue(y)` Returns the expected value.

`Unc.GetFcnValue(y)` Returns the function value.

`Unc.GetStdUnc(y)` Computes the standard uncertainty.

`Unc.GetCoverageInterval(y, p)` Computes the coverage interval.

`Unc.GetMoment(y, n)` Computes the n-th central moment.

`Unc.GetCorrelation([y1 y2 ...])` Computes the correlation matrix.

`Unc.GetCovariance([y1 y2 ...])` Computes the covariance matrix.

`Unc.GetIDof(y)` Computes the inverse degrees of freedom.[2]

`1.0 / Unc.GetIDof(y)` Computes the degrees of freedom.[2]

`Unc.GetJacobi(y)` Returns the sensitivities to the virtual base inputs (with value 0 and uncertainty 1).[2]

`Unc.GetJacobi2(y, x)` Computes the sensitivities of $\mathbf{y}$ to the intermediate results $\mathbf{x}$.[2]

`Unc.GetUncComponent(y, x)` Computes the uncertainty components of $\mathbf{y}$ with respect to $\mathbf{x}$.[2]

[2] `LinProp` uncertainty objects only

6 Storage functions

6.1 Store a computed uncertainty object

`y.BinarySerialize(filepath)` Binary serializes uncertainty object $\mathbf{y}$ to file.

`Misc.Storage.BinarySerialize(y, filepath)` Binary serializes uncertainty object $\mathbf{y}$ to file.

`y.BinarySerializeToByteArray()` Binary serializes uncertainty object $\mathbf{y}$ to byte array.

`Misc.Storage.BinarySerializeToByteArray(y)` Binary serializes uncertainty object $\mathbf{y}$ to byte array.

`y.XmlSerialize(filepath)` XML serializes uncertainty object $\mathbf{y}$ to file.

`Misc.Storage.XmlSerialize(y, filepath)` XML serializes uncertainty object $\mathbf{y}$ to file.

`y.XmlSerializeToString()` XML serializes uncertainty object $\mathbf{y}$ to string.

`Misc.Storage.XmlSerializeToString(y)` XML serializes uncertainty object $\mathbf{y}$ to string.

6.2 Reload a stored uncertainty object

`y.BinaryDeserialize(filepath)` Reloads uncertainty object from binary file.

`Misc.Storage.BinaryDeserialize<T>(filepath)` Reloads uncertainty object from binary file.

`y.BinaryDeserializeFromByteArray(d)` Reloads uncertainty object from byte array.

`Misc.Storage.BinaryDeserializeFromByteArray<T>(d)` Reloads uncertainty object from byte array.

`y.XmlDeserialize(filepath)` Reloads uncertainty object from XML file.

`Misc.Storage.XmlDeserialize<T>(filepath)` Reloads uncertainty object from XML file.

`y.XmlDeSerializeFromString(s)` Reloads uncertainty object from XML string.

`Misc.Storage.XmlDeserializeFromString<T>(s)` Reloads uncertainty object from XML string.