METAS UncLib C# - User Reference V3.0.2

Michael Wollensack

July 2026

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 2019 and it requires the .NET Framework V4.6.2. 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

`LinProp.Misc.Global.DofMode` Set the degrees of freedom mode to WelchSatterthwaite or to StudentT. Default value: WelchSatterthwaite

`LinProp.Misc.Global.FromSamplesMode` Set the from samples mode to Dof or to ExpandInputCovariance. Default value: ExpandInputCovariance

`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.

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

`var x = new UncNumber(value, stdunc, (idof), (id), (dof));` Creates a new real uncertain number with value, standard uncertainty, inverse degrees of freedom (optional), an ID (optional) and a description (optional).

`var x = Unc.RealUncNumber(value, stdunc, (idof), (id), (desc));` Creates a new real uncertain number with value, standard uncertainty, inverse degrees of freedom (optional), an ID (optional) and a description (optional).

`var x = Unc.ComplexUncNumber(value, [covariance], (idof), (id), (desc));` Creates a new complex uncertain number. Covariance size: 2×2. Covariance normalized to $dof = n - 2$.

`var x = Unc.RealUncArray([value], [covariance], (idof), (id), (desc));` Creates a new real uncertain array. Covariance size: $N \times N$. Covariance normalized to $dof = n - N$.

`var x = Unc.ComplexUncArray([value], [covariance], (idof), (id), (desc));` Creates a new complex uncertain array. Covariance size: $2N \times 2N$. Covariance normalized to $dof = n - 2N$.

`var x = Unc.RealUncNumberFromSamples([samples], (id), (desc), (p));` Creates a new real uncertain number from samples with an ID (optional), a description (optional) and a probability (optional). Samples size: n where n is the number of observations.

`var x = Unc.ComplexUncNumberFromSamples([samples], (id), (desc), (p));` Creates a new complex uncertain number from samples with an ID (optional), a description (optional) and a probability (optional). Samples size: n where n is the number of observations. The complex uncertain number contains the correlation between real and imaginary parts.

`var x = Unc.RealUncArrayFromSamples([samples], (id), (desc), (p));` Creates a new real uncertain array from samples with an ID (optional), a description (optional) and a probability (optional). Samples size: $n \times N$ where n is the number of observations and N is the number of dimensions. The real uncertain array contains the correlation between the different entries.

`var x = Unc.ComplexUncArrayFromSamples([samples], (id), (desc), (p));` Creates a new complex uncertain array from samples with an ID (optional), a description (optional) and a probability (optional). Samples size: $n \times N$ where n is the number of observations and N is the number of dimensions. The complex uncertain array contains the correlation between real and the imaginary parts and the different entries.

`var x = Unc.RealUncNumberFromRandomChoices([samples], (id), (desc));` Creates a new real uncertain number from random choices with an ID (optional) and a description (optional). Samples size: n where n is the number of observations.

`var x = Unc.ComplexUncNumberFromRandomChoices([samples], (id), (desc));` Creates a new complex uncertain number from random choices with an ID (optional) and a description (optional). Samples size: n where n is the number of observations. The complex uncertain number contains the correlation between real and imaginary parts.

`var x = Unc.RealUncArrayFromRandomChoices([samples], (id), (desc));` Creates a new real uncertain array from random choices with an ID (optional) and a description (optional). Samples size: $n \times N$ where n is the number of observations and N is the number of dimensions. The real uncertain array contains the correlation between the different entries.

`var x = Unc.ComplexUncArrayFromRandomChoices([samples], (id), (desc));` Creates a new complex uncertain array from random choices with an ID (optional) and a description (optional). Samples size: $n \times N$ where n is the number of observations and N is the number of dimensions. The complex uncertain array contains the correlation between real and the imaginary parts and the different entries.

`var x = new UncNumber(distribution, (id), (desc));` Creates a new real uncertain number from a distribution with an ID (optional) and a description (optional).

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

3.1 Distributions

`var x = new Unc.StandardNormalDistribution();` Creates a normal distribution with $\mu = 0$ and $\sigma = 1$.

`var x = new Unc.NormalDistribution(mu, sigma);` Creates a normal distribution with μ and σ.

`var x = new Unc.StandardUniformDistribution();` Creates an uniform distribution between $a = 0$ and $b = 1$.

`var x = new Unc.UniformDistribution(a, b);` Creates an uniform distribution between a and b.

`var x = new Unc.CurvilinearTrapezoidDistribution(a, b, d);` Creates a curvilinear trapezoid distribution between $a \pm d$ and $b \pm d$.

`var x = new Unc.TrapezoidalDistribution(a, b, beta);` Creates a trapezoidal distribution between a and b with β.

`var x = new Unc.TriangularDistribution(a, b);` Creates a triangular distribution between a and b.

[2] `LinProp` uncertainty objects only

`var x = new Unc.ArcSineDistribution(a, b);` Creates an arc sine distribution between a and b.

`var x = new Unc.ExponentialDistribution(mu);` Creates an exponential distribution with μ.

`var x = new Unc.GammaDistribution(a, b);` Creates a gamma distribution with shape a and scale b.

`var x = new Unc.ChiSquaredDistribution(k);` Creates a chi-squared distribution with degrees of freedom k.

`var x = new Unc.StudentTDistribution(mu, sigma, dof);` Creates a Student T distribution with μ, σ and dof.

`var x = new Unc.StudentTFromSamplesDistribution([samples]);` Creates a Student T distribution from samples.

`var x = new Unc.RandomChoicesFromSamples(seed, [samples]);` Creates random choices from samples with a seed.

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)`
- `Math.Erf(x)`
- `Math.Ellipk(m)`
- `Math.Ellipe(m)`
- `Math.Ellippi(n,m)`
- `Math.InvErf(x)`

4.2 Linear algebra

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

`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 using QR decomposition

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

`LinAlg.GeneralLstSqrSolve(A, Y, V)` General least square solve over determined equation system using QR decomposition

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

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

`LinAlg.Qr(M)` QR decomposition of matrix $\mathbf{M}$

`LinAlg.Svd(M)` Single value decomposition of matrix $\mathbf{M}$

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

[2] `LinProp` uncertainty objects only

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.Gradient(f, x)` Numerical gradient

`NumLib.Fft(v)` Fast Fourier transformation

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

`UncNumLib.Dft(v)` Discrete Fourier transformation[2]

`UncNumLib.Idft(v)` Inverse discrete Fourier transformation[2]

`UncNumerical.NumericalStep(@func, x, dx)` Numerical step[2]

`UncOdeSolver.Solve(y, x, p, eps, @func)` ODE solver[2]

`Optimizer.Start(@func, xStart, p)` Optimizer[2]

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).

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

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

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.

[2] `LinProp` uncertainty objects only

A Physical constants

`Const<UncNumber>` is equal to the newest physical constants `Const2022<UncNumber>`, see subsection A.4.

A.1 CODATA 2014

The following list contains the exact physical constants:

`Const2014<UncNumber>.deltavCs` Hyperfine transition frequency of Cs-133 in Hz

`Const2014<UncNumber>.c0` Speed of light in vacuum in m/s

`Const2014<UncNumber>.mu0` Vacuum magnetic permeability in Vs/Am

`Const2014<UncNumber>.ep0` Vacuum electric permittivity in As/Vm

`Const2014<UncNumber>.Kcd` Luminous efficacy in lm/W

`Const2014<UncNumber>.Mu` Molar mass constant in kg/mol

The following list contains the physical constants with uncertainties:

`Const2014<UncNumber>.G` Newtonian constant of gravitation[3] in $m^3/(kg*s^2)$

`Const2014<UncNumber>.alpha` Fine-structure constant[3]

`Const2014<UncNumber>.Ryd` Rydberg constant[3] in 1/m

`Const2014<UncNumber>.mpsme` Proton-electron mass ratio[3]

`Const2014<UncNumber>.Na` Avogadro constant[3] in 1/mol

`Const2014<UncNumber>.Kj` Josephson constant[3] in Hz/V

`Const2014<UncNumber>.k` Boltzmann constant[3] in J/K

`Const2014<UncNumber>.Rk` von Klitzing constant in Ohm

`Const2014<UncNumber>.e` Elementary charge in C

`Const2014<UncNumber>.h` Planck constant in Js

`Const2014<UncNumber>.me` Electron mass in kg

`Const2014<UncNumber>.mp` Proton mass in kg

`Const2014<UncNumber>.u` Atomic mass constant in kg

`Const2014<UncNumber>.F` Faraday constant in C/mol

`Const2014<UncNumber>.R` Molar gas constant in J/(mol*K)

`Const2014<UncNumber>.eV` Electron volt in J

[3] The correlation matrix of this physical constants is used in METAS UncLib to generate uncertainty objects which are correlated. The other physical constants are computed out of this set and the exact physical constants, e.g.: `Rk = mu0*c0/(2*alpha)` and `e = 2/(Kj*Rk)`.

A.2 CODATA 2014 for conventional electrical units 90

The following list contains the exact physical constants:

`Const2014_90<UncNumber>.deltavCs` Hyperfine transition frequency of Cs-133 in Hz

`Const2014_90<UncNumber>.c0` Speed of light in vacuum in m/s

`Const2014_90<UncNumber>.mu0` Vacuum magnetic permeability in Vs/Am

`Const2014_90<UncNumber>.ep0` Vacuum electric permittivity in As/Vm

`Const2014_90<UncNumber>.Kcd` Luminous efficacy in lm/W

`Const2014_90<UncNumber>.Mu` Molar mass constant in kg/mol

`Const2014_90<UncNumber>.Kj` Conventional value of Josephson constant in Hz/V

`Const2014_90<UncNumber>.Rk` Conventional value of von Klitzing constant in Ohm

`Const2014_90<UncNumber>.e` Elementary charge in C

`Const2014_90<UncNumber>.h` Planck constant in Js

The following list contains the physical constants with uncertainties:

`Const2014_90<UncNumber>.Na` Avogadro constant in 1/mol

`Const2014_90<UncNumber>.F` Faraday constant in C/mol

`Const2014_90<UncNumber>.k` Boltzmann constant in J/K

A.3 CODATA 2018

The following list contains the exact physical constants:

`Const2018<UncNumber>.deltavCs` Hyperfine transition frequency of Cs-133 in Hz

`Const2018<UncNumber>.c0` Speed of light in vacuum in m/s

`Const2018<UncNumber>.h` Planck constant in Js

`Const2018<UncNumber>.e` Elementary charge in C

`Const2018<UncNumber>.k` Boltzmann constant in J/K

`Const2018<UncNumber>.Na` Avogadro constant in 1/mol

`Const2018<UncNumber>.Kcd` Luminous efficacy in lm/W

`Const2018<UncNumber>.Kj` Josephson constant in Hz/V

`Const2018<UncNumber>.Rk` von Klitzing constant in Ohm

`Const2018<UncNumber>.F` Faraday constant in C/mol

`Const2018<UncNumber>.R` Molar gas constant in J/(mol*K)

`Const2018<UncNumber>.eV` Electron volt in J

The following list contains the physical constants with uncertainties:

`Const2018<UncNumber>.G` Newtonian constant of gravitation[4] in $m^3/(kg*s^2)$

`Const2018<UncNumber>.alpha` Fine-structure constant[4]

`Const2018<UncNumber>.mu0` Vacuum magnetic permeability in Vs/Am

`Const2018<UncNumber>.ep0` Vacuum electric permittivity in As/Vm

`Const2018<UncNumber>.Ryd` Rydberg constant[4] in 1/m

`Const2018<UncNumber>.me` Electron mass in kg

`Const2018<UncNumber>.are` Electron relative atomic mass[4]

`Const2018<UncNumber>.arp` Proton relative atomic mass[4]

`Const2018<UncNumber>.mpsme` Proton-electron mass ratio

`Const2018<UncNumber>.mp` Proton mass in kg

`Const2018<UncNumber>.u` Atomic mass constant in kg

`Const2018<UncNumber>.Mu` Molar mass constant in kg/mol

[4]The correlation matrix of this physical constants is used in METAS UncLib to generate uncertainty objects which are correlated. The other physical constants are computed out of this set and the exact physical constants, e.g.: `mu0 = 2*h/(e*e*c0)*alpha` and `ep0 = 1.0/(c0*c0*mu0)`.

A.4 CODATA 2022

The following list contains the exact physical constants:

`Const2022<UncNumber>.deltavCs` Hyperfine transition frequency of Cs-133 in Hz

`Const2022<UncNumber>.c0` Speed of light in vacuum in m/s

`Const2022<UncNumber>.h` Planck constant in Js

`Const2022<UncNumber>.e` Elementary charge in C

`Const2022<UncNumber>.k` Boltzmann constant in J/K

`Const2022<UncNumber>.Na` Avogadro constant in 1/mol

`Const2022<UncNumber>.Kcd` Luminous efficacy in lm/W

`Const2022<UncNumber>.Kj` Josephson constant in Hz/V

`Const2022<UncNumber>.Rk` von Klitzing constant in Ohm

`Const2022<UncNumber>.F` Faraday constant in C/mol

`Const2022<UncNumber>.R` Molar gas constant in J/(mol*K)

`Const2022<UncNumber>.eV` Electron volt in J

The following list contains the physical constants with uncertainties:

`Const2022<UncNumber>.G` Newtonian constant of gravitation[5] in $m^3/(kg*s^2)$

`Const2022<UncNumber>.alpha` Fine-structure constant

`Const2022<UncNumber>.mu0` Vacuum magnetic permeability[5] in Vs/Am

`Const2022<UncNumber>.ep0` Vacuum electric permittivity in As/Vm

`Const2022<UncNumber>.Ryd` Rydberg constant[5] in 1/m

`Const2022<UncNumber>.me` Electron mass in kg

`Const2022<UncNumber>.are` Electron relative atomic mass[5]

`Const2022<UncNumber>.arp` Proton relative atomic mass[5]

`Const2022<UncNumber>.mpsme` Proton-electron mass ratio

`Const2022<UncNumber>.mp` Proton mass in kg

`Const2022<UncNumber>.u` Atomic mass constant in kg

`Const2022<UncNumber>.Mu` Molar mass constant in kg/mol

[5]The correlation matrix of this physical constants is used in METAS UncLib to generate uncertainty objects which are correlated. The other physical constants are computed out of this set and the exact physical constants, e.g.: `alpha = e*e*c0/(2*h)*mu0` and `ep0 = 1.0/(c0*c0*mu0)`.