**Uncertainty Toolbox**

**User’s Manual**

Slimane Aoudjit

Wolfgang Granig

Hubert Zangl

Dailys Arronde Pérez

**Uncertainty Toolbox Version V0.91, 22.05.2019**

# Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Author** | **Approved** | **Version** |
| Initial version | Granig |  | V0.1 |
| Version 0.9 | Aoudjit |  | V0.9 |
| Version 0.91 | Arronde |  | V0.91 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Content

[1. Revision history 2](#_Toc24583313)

[2. Content 3](#_Toc24583314)

[3. Disclaimer 8](#_Toc24583315)

[4. General description 8](#_Toc24583316)

[5. Properties of the Uncertainty Class 9](#_Toc24583317)

[5.1. Linearization 9](#_Toc24583318)

[5.1.1 The name Property 9](#_Toc24583319)

[5.1.2 The value Property 9](#_Toc24583320)

[5.1.3 The std\_unc Property 9](#_Toc24583321)

[5.1.4 The dep Property 9](#_Toc24583322)

[5.1.5 The grad Property 10](#_Toc24583323)

[5.1.6 The OutDataTypeStr Property 10](#_Toc24583324)

[5.1.7 The rel\_obj Property 10](#_Toc24583325)

[5.1.8 The rel\_mat Property 10](#_Toc24583326)

[5.1.9 The r Property 11](#_Toc24583327)

[5.1.10 The img Property 11](#_Toc24583328)

[5.2. Monte Carlo(MC) 11](#_Toc24583329)

[5.2.1 The values property 11](#_Toc24583330)

[5.3. Unscented Transformation (UT) 11](#_Toc24583331)

[5.3.1 The nom\_value property 11](#_Toc24583332)

[5.3.2 The dep property 11](#_Toc24583333)

[5.3.3 The sigma property 12](#_Toc24583334)

[6. Creation of Uncertainty Objects 12](#_Toc24583335)

[6.1. Syntax 1: unc(value,std\_unc,name) 12](#_Toc24583336)

[6.2. Syntax 2: unc(value,std\_unc,name) 12](#_Toc24583337)

[6.2.1 Syntax 2.1: unc(value, std\_unc , name ) 13](#_Toc24583338)

[6.3. Syntax 3: unc( unc\_objects , CX , 'cov') 13](#_Toc24583339)

[6.4. Syntax 4: unc( unc\_objects , CX , 'corr') 13](#_Toc24583340)

[6.5. Syntax 5: unc( value , std\_unc ) 14](#_Toc24583341)

[6.6. Syntax 6: unc( value , name ) 14](#_Toc24583342)

[6.7. Syntax 7: unc( value , name ) 15](#_Toc24583343)

[6.8. Syntax 8: unc( value , CX ) 15](#_Toc24583344)

[6.9. Syntax 9: unc( unc\_objects , CX ) 16](#_Toc24583345)

[6.10. Syntax 10: unc( value ) 16](#_Toc24583346)

[6.11. Syntax 11: unc( unc\_object ) 16](#_Toc24583347)

[6.12. Syntax 12: unc( ) 17](#_Toc24583348)

[7. Setting Covariance and Correlation 17](#_Toc24583349)

[7.1. Setting covariance and Correlation for Uncertainty Objects 17](#_Toc24583350)

[7.1.1 Example 17](#_Toc24583351)

[8. Covariance and Correlation Matrices 18](#_Toc24583352)

[8.1. Recovery of the Covariance Matrix of a Set of Uncertainty Objects 19](#_Toc24583353)

[8.1.1 Example 1 19](#_Toc24583354)

[8.1.2 Example 2 19](#_Toc24583355)

[8.2. Recovery of the Correlation Matrix of a Set of Uncertainty Objects 21](#_Toc24583356)

[8.2.1 Example 1 21](#_Toc24583357)

[8.2.2 Example 2 21](#_Toc24583358)

[8.3. Transforming a Covariance Matrix into a Correlation Matrix 22](#_Toc24583359)

[8.3.1 Example 22](#_Toc24583360)

[8.4. Transforming a Correlation Matrix into a Covariance Matrix 23](#_Toc24583361)

[8.4.1 Example 23](#_Toc24583362)

[9. Uncertainty Propagation in Real-Valued Quantities 24](#_Toc24583363)

[9.1. Addition of Real-Valued Quantities 24](#_Toc24583364)

[9.1.1 Example 1 24](#_Toc24583365)

[9.1.2 Example 2 24](#_Toc24583366)

[9.2. Addition of a Real-Valued Quantity and a Constant Number 25](#_Toc24583367)

[9.2.1 Example 25](#_Toc24583368)

[9.3. Subtraction of Real-Valued Quantities 26](#_Toc24583369)

[9.3.1 Example 1 26](#_Toc24583370)

[9.3.2 Example 2 26](#_Toc24583371)

[9.4. Subtraction of a Real-Valued Quantity and a Constant Number 27](#_Toc24583372)

[9.4.1 Example 27](#_Toc24583373)

[9.5. Product of Real-Valued Quantities 28](#_Toc24583374)

[9.5.1 Example 1 28](#_Toc24583375)

[9.5.2 Example 2 29](#_Toc24583376)

[9.6. Product of a Real-Valued Quantity and a Constant Number 29](#_Toc24583377)

[9.7. Ratio of Real-Valued Quantities 30](#_Toc24583378)

[9.7.1 Example 1 30](#_Toc24583379)

[9.7.2 Example 2 31](#_Toc24583380)

[9.8. Ratio of a Real-Valued Quantity and a Constant Number 31](#_Toc24583381)

[9.8.1 Example 1 31](#_Toc24583382)

[9.8.2 Example 2 32](#_Toc24583383)

[9.9. Real Logarithm Functions 32](#_Toc24583384)

[9.9.1 Real Natural Logarithm Function log() 32](#_Toc24583385)

[9.9.2 Real Decimal Logarithm Function log10() 32](#_Toc24583386)

[9.10. Real Exponential Function exp() 33](#_Toc24583387)

[Example: 33](#_Toc24583388)

[9.11. Real Trigonometric Functions 33](#_Toc24583389)

[9.11.1 Real Sine Function sin() 33](#_Toc24583390)

[9.11.2 Real Cosine Function cos() 33](#_Toc24583391)

[9.11.3 Real Tangent Function tan() 34](#_Toc24583392)

[9.11.4 Real Cotangent Function cot() 34](#_Toc24583393)

[9.11.5 Inverse Sine Function asin() 34](#_Toc24583394)

[9.11.6 Inverse Cosine Function acos() 34](#_Toc24583395)

[9.11.7 Inverse Tangent Function atan() 35](#_Toc24583396)

[9.11.8 Inverse Cotangent Function acot() 35](#_Toc24583397)

[9.11.9 Four-Quadrant Inverse Tangent Function atan2() 35](#_Toc24583398)

[9.12. Real Hyperbolic Functions 35](#_Toc24583399)

[9.12.1 Real Hyperbolic Sine Function sinh() 35](#_Toc24583400)

[9.12.2 Real Hyperbolic Cosine Function cosh() 36](#_Toc24583401)

[9.12.3 Real Hyperbolic Tangent Function tanh() 36](#_Toc24583402)

[9.12.4 Real Hyperbolic Cotangent Function coth() 36](#_Toc24583403)

[9.12.5 Inverse Hyperbolic Sine Function asinh() 36](#_Toc24583404)

[9.12.6 Inverse Hyperbolic Cosine Function acosh() 37](#_Toc24583405)

[9.12.7 Inverse Hyperbolic Tangent Function atanh() 37](#_Toc24583406)

[9.13. Square Root Function sqrt() 37](#_Toc24583407)

[9.14. Covariance and Correlation Propagation 37](#_Toc24583408)

[9.14.1 Example 1 37](#_Toc24583409)

[9.14.2 Example 2 38](#_Toc24583410)

[9.14.3 Example 3 39](#_Toc24583411)

[9.15. Karhunen-Loeve Decomposition 40](#_Toc24583412)

[9.15.1 Example 1 40](#_Toc24583413)

[9.15.2 Example 2 41](#_Toc24583414)

[9.15.3 Example 3 42](#_Toc24583415)

[10. Uncertainty Propagation in Complex-Valued Quantities 44](#_Toc24583416)

[10.1. Creation of Complex Uncertainty Objects 44](#_Toc24583417)

[10.1.1 Example 1 44](#_Toc24583418)

[10.1.2 Example 2 44](#_Toc24583419)

[10.1.3 Example 3 45](#_Toc24583420)

[10.1.4 Example 4 45](#_Toc24583421)

[10.2. Addition of Complex Uncertainty Objects 45](#_Toc24583422)

[10.2.1 Example 45](#_Toc24583423)

[10.3. Subtraction of Complex Uncertainty Objects 46](#_Toc24583424)

[10.3.1 Example 46](#_Toc24583425)

[10.4. Product of Complex Uncertainty Objects 46](#_Toc24583426)

[10.4.1 Example 46](#_Toc24583427)

[10.5. Mixed Product of Complex and Real Uncertainty Objects 47](#_Toc24583428)

[10.5.1 Example 47](#_Toc24583429)

[10.6. Ratio of Complex Uncertainty Objects 47](#_Toc24583430)

[10.6.1 Example 47](#_Toc24583431)

[10.7. Mixed Ratio of Complex and Real Uncertainty Objects 48](#_Toc24583432)

[10.7.1 Example 1 48](#_Toc24583433)

[10.7.2 Example 2 49](#_Toc24583434)

[10.8. Absolute Value of a Complex Object 49](#_Toc24583435)

[10.8.1 Example 49](#_Toc24583436)

[10.9. Conjugate of a Complex Object 49](#_Toc24583437)

[10.9.1 Example 49](#_Toc24583438)

[10.10. Complex Functions 50](#_Toc24583439)

[10.10.1 Complex Logarithmic Functions 50](#_Toc24583440)

[1) Complex Natural Logarithm Function log() 50](#_Toc24583441)

[Example: 50](#_Toc24583442)

[2) Complex Decimal Logarithm Function log10() 51](#_Toc24583443)

[Example: 51](#_Toc24583444)

[10.10.2 Complex Exponential Function exp() 51](#_Toc24583445)

[Example: 51](#_Toc24583446)

[10.10.3 Complex Trigonometric Functions 52](#_Toc24583447)

[1) Complex Cosine Function cos() 52](#_Toc24583448)

[Example: 52](#_Toc24583449)

[2) Complex Sine Function sin() 53](#_Toc24583450)

[Example: 53](#_Toc24583451)

[3) Complex Tangent Function tan() 53](#_Toc24583452)

[Example: 53](#_Toc24583453)

[4) Complex Cotangent Function cot() 54](#_Toc24583454)

[Example: 54](#_Toc24583455)

[5) Complex Inverse Sine Function asin() 54](#_Toc24583456)

[Example: 54](#_Toc24583457)

[6) Complex Inverse Cosine Function acos() 54](#_Toc24583458)

[Example: 54](#_Toc24583459)

[7) Complex Inverse Tangent Function atan() 55](#_Toc24583460)

[Example: 55](#_Toc24583461)

[8) Complex Inverse Cotangent Function acot() 55](#_Toc24583462)

[Example: 55](#_Toc24583463)

[10.10.4 Complex Hyperbolic Functions 55](#_Toc24583464)

[1) Complex Hyperbolic Sine Function sinh() 55](#_Toc24583465)

[Example: 55](#_Toc24583466)

[2) Complex Hyperbolic Cosine Function cosh() 56](#_Toc24583467)

[Example: 56](#_Toc24583468)

[3) Complex Hyperbolic Tangent Function tanh() 57](#_Toc24583469)

[Example: 57](#_Toc24583470)

[4) Complex Hyperbolic Cotangent Function coth() 57](#_Toc24583471)

[Example: 57](#_Toc24583472)

[5) Complex Inverse Hyperbolic Sine Function asinh() 58](#_Toc24583473)

[Example: 58](#_Toc24583474)

[6) Complex Inverse Hyperbolic Cosine Function acosh() 58](#_Toc24583475)

[Example: 58](#_Toc24583476)

[7) Complex Inverse Hyperbolic Tangent Function atanh() 58](#_Toc24583477)

[Example: 58](#_Toc24583478)

[10.11. Complex Square Root Function sqrt() 59](#_Toc24583479)

[10.12. Covariance and Correlation Calculation 59](#_Toc24583480)

[10.12.1 Covariance and Correlation between Real and Imaginary Parts of a Complex-Valued Quantity 59](#_Toc24583481)

[Example: 59](#_Toc24583482)

[10.12.2 Covariance and Correlation between Real and Imaginary Parts of more than One Complex-Valued Quantity 60](#_Toc24583483)

[10.12.3 Covariance and Correlation between Two Complex-Valued Quantities 61](#_Toc24583484)

[10.12.4 Covariance and Correlation between a Complex-Valued Quantity and a Real Object 62](#_Toc24583485)

[10.13. Karhunen-Loeve Decomposition 63](#_Toc24583486)

[10.13.1 Karhunen-Loeve Decomposition of One Complex Object 63](#_Toc24583487)

[10.13.2 Karhunen-Loeve Decomposition of more than One Complex Object 64](#_Toc24583488)

[10.13.3 Karhunen-Loeve Decomposition of Mixed Complex and Real Objects 65](#_Toc24583489)

[11. Special Operations on Arrays 66](#_Toc24583490)

[11.1. Determinant 66](#_Toc24583491)

[11.1.1 Example 66](#_Toc24583492)

[11.2. Inverse Matrix 67](#_Toc24583493)

[11.2.1 Example 67](#_Toc24583494)

[11.3. Diagonal Elements of a Matrix 68](#_Toc24583495)

[11.3.1 Example 68](#_Toc24583496)

[11.4. Trace of a Matrix 68](#_Toc24583497)

[11.4.1 Example 68](#_Toc24583498)

[11.5. Largest Element in a 1xn or nx1 Array 69](#_Toc24583499)

[11.5.1 Example 69](#_Toc24583500)

[11.6. Smallest Element in a 1xn or nx1 Array 69](#_Toc24583501)

[11.6.1 Example 69](#_Toc24583502)

[11.7. Sum of Array Elements 70](#_Toc24583503)

[11.7.1 Example 70](#_Toc24583504)

[11.8. Cumulative Sum of Array Elements 70](#_Toc24583505)

[11.8.1 Example 70](#_Toc24583506)

[11.9. Jacobian Matrix 71](#_Toc24583507)

[11.9.1 Example 71](#_Toc24583508)

[11.10. Complex Conjugate Transpose 72](#_Toc24583509)

[11.10.1 Example 72](#_Toc24583510)

[12. Display of Uncertainty Contributions 72](#_Toc24583511)

[12.1. Example 73](#_Toc24583512)

[13. Comparison of Uncertainty Objects 73](#_Toc24583513)

[13.1. The greater than Operator 73](#_Toc24583514)

[13.1.1 Comparison of two Uncertainty Objects 73](#_Toc24583515)

[Example: 73](#_Toc24583516)

[13.1.2 Comparison of an Uncertainty Object with a Constant Number 74](#_Toc24583517)

[Example: 75](#_Toc24583518)

[13.2. The less than Operator 75](#_Toc24583519)

[13.2.1 Comparison of two Uncertainty Objects 75](#_Toc24583520)

[Example: 76](#_Toc24583521)

[13.2.2 Comparison of an Uncertainty Object with a Constant Number 76](#_Toc24583522)

[Example: 77](#_Toc24583523)

[14. Generation of Uncertainty Objects from Data Files 77](#_Toc24583524)

[14.1. Working with Excel Files 77](#_Toc24583525)

[14.1.1 Syntax 1: read\_data( file\_name , sheet\_name , range , 'r/c') 77](#_Toc24583526)

[14.1.2 Syntax 2:read\_data(file\_name , sheet\_name , mean\_val , std\_unc , names) 78](#_Toc24583527)

[14.1.3 Syntax 3: read\_data( file\_name , sheet\_name , mean\_val , std\_unc ) 79](#_Toc24583528)

[14.2. Working with CSV and DAT Files 80](#_Toc24583529)

[14.2.1 Syntax 1: read\_data( file\_name , 'r/c') 80](#_Toc24583530)

[14.2.2 Syntax 2: read\_data( file\_name , R , C , [R C R2 C2] , 'r/c' ) 82](#_Toc24583531)

[15. References 82](#_Toc24583532)

[16. Index of Commands 84](#_Toc24583533)

# Disclaimer

THE SAT/IST/AAU MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

# General description

This uncertainty toolbox was developed in order to enhance the widespread use of uncertainty calculations in measurement science that is directly based on the utilization of software tools. The developed material was initially based on the GUM (Guide to the expression of uncertainty in measurement [1]) method. In this approach, here called Linearization, the measurement model is linearized, and the standard deviation of the output is estimated by applying the law of propagation of uncertainty. In problems involving non-linear systems, Monte Carlo (MC) and Unscented Transformation (UT) represent more suitable methods for the evaluation of uncertainty in measurements. The former, MC method addressed in [2], is a practical alternative to the GUM, when linearization of the model provides and inadequate representation and to estimate the probability density function for the output, when it departs appreciably from the Gaussian distribution. The latter represents a new uncertainty analysis approach based on the principle of the UT, explained in [3], which also provides an adequate alternative to estimate the output quantity and its associated standard uncertainty in non-linear systems and could involve less calculation than MC, in many cases.

The three different approaches aforementioned: Linearization, MC and UT are implemented in this Toolbox, providing a very intuitive way to switch between them and allowing comparison of the results in a very easy fashion.

Additionally, the concept of this toolbox goes beyond the scope of measurement uncertainty quantification demonstrating that it is also useful for system analysis and optimization. [4]

**Key Note:** Most of the examples in this manual are implemented using the linearization approach. For using MC and UT methods, replace unc by unc\_t and unc\_ut respectively. To switch between the methods, just include *unc = @unc\_t* at the beginning of the script for MC and *unc = @unc\_ut* for UT.

There are some functions that are available just in one of the approaches. Such cases will be highlighted along this manual.

# Properties of the Uncertainty Class

The uncertainty class has the following properties:

## Linearization

### The name Property

Name of the uncertainty object. If x is an uncertainty object, then the *name* property of x can be accessed using the command:

x.name;

It can also be accessed using the method gmn():

gmn(x);

Names must be assigned to uncertainty objects in order to be able to perform covariance and correlation calculations.

### The value Property

Mean value of the uncertainty object. If x is an uncertainty object, then the value property of x can be accessed using the command:

x.value;

It can also be accessed using the method gmv():

gmv(x);

### The std\_unc Property

The standard uncertainty of the uncertainty object. If x is an uncertainty object, then the std\_unc property of x can be accessed using the command:

x.std\_unc;

It can also be accessed using the method gmu():

gmu(x);

### The dep Property

Array containing all the uncertainty objects on which a given uncertainty object depends. If x is an uncertainty object, then the *dep* property of x can be accessed using the command:

x.dep;

It can also be accessed using the method disp\_dep():

disp\_dep(x);

The method disp\_dep() displays the objects on which x depends as well as the assigned names of those objects.

### The grad Property

Array containing all the gradient values of the uncertainty object computed at the mean values. If x is an uncertainty object, then the *grad* property of x can be accessed using the command:

x.grad;

### The OutDataTypeStr Property

The data type of the uncertainty object. If x is an uncertainty object, then the *OutDataTypeStr* property of x can be accessed using the command:

x.OutDataTypeStr;

### The rel\_obj Property

Array containing all the uncertainty objects with which a given uncertainty object is correlated. If x is an uncertainty object, then the *rel\_obj* property of x can be accessed using the command:

x.rel\_obj;

### The rel\_mat Property

A 2-row matrix which contains covariance and correlation values of the uncertainty object with respect to the uncertainty objects, which are found in the *rel\_obj* property. The first row contains covariances, whereas the second row contains correlation coefficients.

If x is an uncertainty object, then the *rel\_mat* property of x can be accessed using the command:

x.rel\_mat;

If x is correlated with n other uncertainty objects, then:

To access covariance values, use the command:

x.rel\_mat(1,i);

where i=1,2,…,n.

To access correlation values, use the command:

x.rel\_mat(2,i);

where i=1,2,…,n.

### The r Property

Real part of the uncertainty object. If x is an uncertainty object, then the *r* property of x can be accessed using the command:

x.r;

It can also be accessed using the method *real():*

real(x)

### The img Property

Imaginary part of the uncertainty object. If x is an uncertainty object, then the *img* property of x can be accessed using the command:

x.img;

It can also be accessed using the method *imag():*

imag(x)

## Monte Carlo(MC)

### The values property

Values stores the samples generated when defining the uncertainty object using MC method. If x is an uncertainty object, then the values property of x can be accessed using the command:

x.values;

The samples are generated using random sampling techniques. The number of samples is set to 10000.

## Unscented Transformation (UT)

### The nom\_value property

Nominal value of the uncertainty object. If x is an uncertainty object defined using UT method, then the nom\_value property of x can be accessed using the command:

x.nom\_value;

### The dep property

Array containing all the uncertainty objects on which a given uncertainty object depends. If x is an uncertainty object defined using UT method, then the *dep* property of x can be accessed using the command:

x.dep;

### The sigma property

A 2-row matrix which contains the sigma points of the uncertainty object.

The number of columns is related with the uncertainty objects which are found in the *dep* property. If x is an uncertainty object defined using UT method, then the *sigma* property of x can be accessed using the command:

x.sigma;

If x is dependent on n uncertainty objects, then to access sigma points values, use the command:

x.sigma(1,i);

where i=1,2,…,n.

# Creation of Uncertainty Objects

To create an uncertain variable, use one of the following syntax:

## Syntax 1: unc(value,std\_unc,name)

The input arguments are:

*value*: 1x1 double array corresponding to the property *value*

*std\_unc*: 1x1 double array corresponding to the property *std\_unc*

*name*: 1x1 character array corresponding to the property *name*

**Example:**

x1=unc(5 , 3.1623,'x1');

% x1 =

%

% 5.0(3.2)

## Syntax 2: unc(value,std\_unc,name)

The input arguments are:

*value*: nxm double array corresponding to the property *value*

*std\_unc*: nxm double array corresponding to the property *std\_unc*

*name*: nxm cell array corresponding to the property *name*

**Example:**

X=unc([ 1 2 ; 3 4 ],[ 0.1 0.2 ; 0.3 0.4 ],{ 'x1' 'x2' ; 'x3' 'x4' });

%

% X =

%

% 1.00(10) 2.00(20)

% 3.00(30) 4.00(40)

### Syntax 2.1: unc(value, std\_unc , name )

The input arguments are:

*value*: nxm double array corresponding to the property *value*

*std\_unc*: nxm double array corresponding to the property *std\_unc*

*name*: 1x1 character array corresponding to the property *name*

(In case it is not required to assign a different name to each element of the array)

**Example:**

X=unc([ 1 2 ; 3 4 ],[ 0.1 0.2 ; 0.3 0.4 ], 'X');

%

% X =

%

% 1.00(10) 2.00(20)

% 3.00(30) 4.00(40)

**Note:** Syntaxes 1, 2 and 2.1 can be used with MC and UT methods replacing unc by unc\_t and unc\_ut respectively. In this case, the name assigned to the uncertainty object will be ignored. Remember that MC and UT approaches do not have name property.

## Syntax 3: unc( unc\_objects , CX , 'cov')

The input arguments are:

*unc\_objects*: 1xn array of real uncertainty objects

*CX*: nxn double array

*‘cov’*: indicates that *CX* is a covariance matrix

**Example:**

x1=unc(7.8333, 3.7103 , 'x1');

x2=unc(3.5000, 1.8708 , 'x2');

x3=unc(12.6667, 4.5898 ,'x3');

CX=[13.7667 6.9000 -16.8667 ; 6.9000 3.4999 -8.4000 ; -16.8667 -8.4000 21.0663];

X=unc([x1 x2 x3] , CX ,'cov');

% X =

%

% 7.8(3.7) 3.5(1.9) 12.7(4.6)

## Syntax 4: unc( unc\_objects , CX , 'corr')

The input arguments are:

unc\_objects: 1xn array of real uncertainty objects

CX: nxn double array

‘corr’: indicates that CX is a correlation matrix

**Example:**

x1=unc(7.8333, 3.7103 , 'x1' );

x2=unc(3.5000, 1.8708 , 'x2' );

x3=unc(12.6667, 4.5898 , 'x3' );

CX=[1.0000 0.9940 -0.9904 ; 0.9940 1.0000 -0.9782 ; -0.9904 -0.9782 1.0000];

X=unc([x1 x2 x3] ,CX ,'corr');

% X =

%

% 7.8(3.7) 3.5(1.9) 12.7(4.6)

Note that CX must be symmetric and positive semidefinite, otherwise an error message is shown.

**Note:** Syntaxes 3 and 4 are not available with MC and UT methods.

## Syntax 5: unc( value , std\_unc )

The input arguments are:

value: nxm double array corresponding to the property *value*

std\_unc: nxm double array corresponding to the property *std\_unc*

**Example:**

X=unc([ 1 2 ; 3 4 ],[ 0.1 0.2 ; 0.3 0.4 ]);

% X =

%

% 1.00(10) 2.00(20)

% 3.00(30) 4.00(40)

## Syntax 6: unc( value , name )

The input arguments are:

*value*: 1x1 double array corresponding to the property *value*

*name*: 1x1 character array corresponding to the property *name*

**Example:**

x=unc( 1 , 'x' );

% x =

%

% 1(0)

## Syntax 7: unc( value , name )

The input arguments are:

value: nxm double array corresponding to the property *value*

name: nxm cell array corresponding to the property *name*

**Example:**

X=unc([ 1 2 ; 3 4 ] , { 'x1' 'x2' ; 'x3' 'x4' } );

% X =

%

% 1(0) 2(0)

% 3(0) 4(0)

**Note:** Syntaxes 6 and 7 can be used with MC and UT methods replacing unc by unc\_t and unc\_ut respectively. In this case, the name assigned to the uncertainty object will be ignored. Remember that MC and UT approaches don’t have name property.

## Syntax 8: unc( value , CX )

The input arguments are:

value: nxm double array corresponding to the property *value*

CX: (nxm)x(nxm) double array which corresponds to the covariance matrix of the elements of the array *value*.

**Example:**

values=[ 1 2 ; 3 4 ];

CX = [ 0.1178 0.1863 0.2082 0.2661; 0.1863 0.5105 0.1322 0.4092; 0.2082 0.1322 0.7141 0.6675; 0.2661 0.4092 0.6675 0.8112];

X=unc( values , CX );

% X =

%

% 1.00(34) 2.00(71) 3.00(85) 4.00(90)

Note that CX must be symmetric and positive semidefinite, otherwise an error message is shown.

## Syntax 9: unc( unc\_objects , CX )

The input arguments are:

*unc\_objects*: 1xn array of real uncertainty objects

*CX*: (nxm)x(nxm) double array which corresponds to the covariance matrix of the elements of the array *unc\_objects*.

**Example:**

x1=unc(7.8333, 3.7103 , 'x1' );

x2=unc(3.5000, 1.8708 , 'x2' );

x3=unc(12.6667, 4.5898 , 'x3' );

CX=[13.7667 6.9000 -16.8667 ; 6.9000 3.5000 -8.4000 ; -16.8667 -8.4000 21.0667];

X=unc([x1 x2 x3] , CX );

% X =

%

% 7.8(3.7) 3.5(1.9) 12.7(4.6)

**Note:** Syntaxes 8 and 9 are not available with MC and UT methods.

## Syntax 10: unc( value )

The input argument is:

*value*: nxm double array corresponding to the property *value*

**Example:**

a = unc([ 1 1 2 ; 3 5 8 ; 13 21 34 ]);

% a =

%

% 1(0) 1(0) 2(0)

% 3(0) 5(0) 8(0)

% 13(0) 21(0)

## Syntax 11: unc( unc\_object )

The input argument is:

*unc\_object*: 1x1 array of uncertainty objects

**Example:**

a = unc( 13 , 0.5 );

b = unc( a );

% b =

%

% 13.00(50)

## Syntax 12: unc( )

Empty input argument.

**Example:**

x = unc();

% x =

%

% 0(0)

# Setting Covariance and Correlation

## Setting covariance and Correlation for Uncertainty Objects

### Example

Consider three real objects x1, x2, and x3:

x1=unc(0.8700, 0.7103 , 'x1' );

x2=unc(0.9400, 0.5337 , 'x2' );

x3=unc(2.1130, 0.5759 , 'x3' );

Enter the corresponding covariance matrix CX:

CX=[ 0.5046 -0.0864 -0.1562 ; -0.0864 0.2849 0.0164 ; -0.1562 0.0164 0.3317];

unc([x1 x2 x3] , CX ,'cov');

Consider two output real objects y1 and y2 defined by:

y1=x1 + x2 ;

y2=x3-x1;

In this case, covariance and correlation between the outputs and any other objects can be checked as follow:

covr(x1,y1);

covr(y1,x3);

covr(y1,y2);

% ans =

%

% 0.4181

%

% ans =

%

% -0.1398

%

% ans =

%

% -0.5579

Correlation or covariance coefficient between two objects that are no correlated with any others, can be set using the method set\_correl or set\_cov. For example:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

set\_correl( a , b , 0.5 );

To check if the correlation was correctly set, use the function correl, as follow:

correl(a,b);

% ans =

%

% 0.5000

covr(a,b);

% ans =

%

% 0.3323

Note: It is important to highlight that Covariance and Correlation between two or more uncertainty objects should be assigned using the syntaxes 3, 4, 8 and 9. The functions set\_cov and set\_correl can be used only if the variables involved aren’t correlated with any other uncertain variable.

**Note:** Functions described in section 7 are not available with MC and UT methods.

# Covariance and Correlation Matrices

## Recovery of the Covariance Matrix of a Set of Uncertainty Objects

### Example 1

Consider three real objects x1, x2, and x3:

x1=unc(1.8100,0.7852,'x1');

x2=unc(1.9100,0.8465,'x2');

x3=unc(2.2900,0.8569,'x3');

Enter the corresponding covariance matrix CX:

CX = [ 0.6166 0.6154 0.6057 ; ...

0.6154 0.7166 0.6934 ; ...

0.6057 0.6934 0.7343 ];

unc([x1 x2 x3] , CX ,'cov');

You can check that the method get\_cov\_mat() can properly recover the covariance matrix CX:

[ CX ] = get\_cov\_mat([x1 x2 x3]);

% CX =

%

% 0.6165 0.6154 0.6057

% 0.6154 0.7166 0.6934

% 0.6057 0.6934 0.7343

%

### Example 2

Consider three real objects a, b, and c:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

c=unc(2.2900,0.8569,'c');

Enter the corresponding covariance matrix CX:

CX = [ 0.6166 0.6154 0.6057 ; ...

0.6154 0.7166 0.6934 ; ...

0.6057 0.6934 0.7343 ];

unc([a b c] , CX ,'cov');

Create two complex objects z1 and z2:

z1=complex(a,b);

z2=complex(b,c);

% z1 =

%

% 1.81(79) + 1.91(85) \* i

%

% z2 =

%

% 1.91(85) + 2.29(86) \* i

Create the array Z of complex objects:

Z=[z1 z2];

% Z =

%

% 1.81(79) + 1.91(85) \* i 1.91(85) + 2.29(86) \* i

By recovering the covariance matrix of the array Z, we can get two possible results

1. [ CZ ] = get\_cov\_mat(Z);

% CZ =

%

% 0.6166 0.6154 0.6154 0.6057

% 0.6154 0.7166 0.7166 0.6934

% 0.6154 0.7166 0.7166 0.6934

% 0.6057 0.6934 0.6934 0.7343%

%

Where CZ is the covariance matrix associated with the vector containing the real and imaginary components of Z, i.e. a vector Z’ is created out of Z, where Z’ = [real(z1) imag(z1) real(z2) imag(z2)] and CZ is the covariance matrix of Z’. In general, if vector Zϵℝn, then Z’ϵℝ2n and CZ is [2n x 2n] matrix of real elements.

1. [ CZ, CZ\_complex] = get\_cov\_mat(Z);

% CZ =

%

% 0.6166 0.6154 0.6154 0.6057

% 0.6154 0.7166 0.7166 0.6934

% 0.6154 0.7166 0.7166 0.6934

% 0.6057 0.6934 0.6934 0.7343

%

%

% CZ\_complex =

%

% 1.3331 + 0.0000i 1.3088 + 0.1109i

% 1.3088 - 0.1109i 1.4508 + 0.0000i

Where CZ\_complex is a [n x n] complex matrix, resulting from CZ\_complex.

## Recovery of the Correlation Matrix of a Set of Uncertainty Objects

### Example 1

Similarly to the above examples, we can recover the correlation matrix of a set of uncertainty objects using the method *get\_cor\_mat().*

Consider three real objects x1, x2, and x3:

x1=unc(1.8100,0.7852,'x1');

x2=unc(1.9100,0.8465,'x2');

x3=unc(2.2900,0.8569,'x3');

Enter the corresponding covariance matrix CX:

CX = [ 0.6166 0.6154 0.6057 ; ...

0.6154 0.7166 0.6934 ; ...

0.6057 0.6934 0.7343 ];

unc([x1 x2 x3] , CX ,'cov');

Getting the correlation matrix

[CorX ] = get\_cor\_mat([x1 x2 x3]);

% CorX =

%

% 1.0000 0.9259 0.9002

% 0.9259 1.0000 0.9559

% 0.9002 0.9559 1.0000

%

### Example 2

Consider three real objects a, b, and c:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

c=unc(2.2900,0.8569,'c');

Enter the corresponding covariance matrix CX:

CX = [ 0.6166 0.6154 0.6057 ; ...

0.6154 0.7166 0.6934 ; ...

0.6057 0.6934 0.7343 ];

unc([a b c] , CX ,'cov');

Create two complex objects z1 and z2:

z1=complex(a,b);

z2=complex(b,c);

% z1 =

%

% 1.81(79) + 1.91(85) \* i

%

% z2 =

%

% 1.91(85) + 2.29(86) \* i

Create the array Z of complex objects:

Z=[z1 z2];

% Z =

%

% 1.81(79) + 1.91(85) \* i 1.91(85) + 2.29(86) \* i

In this case, we can recover the correlation matrix of Z by

[ CZ\_cor ] = get\_cor\_mat(Z);

% CZ\_cor =

%

% 1.0000 0.9259 0.9259 0.9002

% 0.9259 1.0000 1.0000 0.9559

% 0.9259 1.0000 1.0000 0.9559

% 0.9002 0.9559 0.9559 1.0000

%

Where CZ\_cor is the correlation matrix associated with the vector containing the real and imaginary components of Z, i.e. a vector Z’ is created out of Z, where Z’ = [real(z1) imag(z1) real(z2) imag(z2)] and CZ\_cor is the correlation matrix of Z’. In general, if vector Zϵℝn, then Z’ϵℝ2n and CZ\_cor is [2n x 2n] matrix of real elements.

## Transforming a Covariance Matrix into a Correlation Matrix

### Example

Consider three real objects a, b, and c:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

c=unc(2.2900,0.8569,'c');

Enter the corresponding covariance matrix CX:

CX\_cov = [ 0.6166 0.6154 0.6057 ; ...

0.6154 0.7166 0.6934 ; ...

0.6057 0.6934 0.7343 ];

unc([a b c] , CX\_cov ,'cov');

In order to transform the covariance matrix CX\_cov into its corresponding correlation matrix CX\_corr, the method cov\_into\_cor\_mat(X,CX\_cov) is used, where:

X: array of uncertainty objects

CX\_cov: the covariance matrix of the elements of X

Therefore, using the cov\_into\_cor\_mat() for this example yields:

CX\_corr = cov\_into\_cor\_mat([a b c],CX\_cov);

% CX\_corr =

%

% 1.0001 0.9259 0.9002

% 0.9259 1.0001 0.9559

% 0.9002 0.9559 1.0000

## Transforming a Correlation Matrix into a Covariance Matrix

### Example

Consider three objects x1, x2, and x3:

x1=unc(7.8333, 3.7103 , 'x1' );

x2=unc(3.5000, 1.8708 , 'x2' );

x3=unc(12.6667, 4.5898 , 'x3' );

Enter the corresponding correlation matrix CX\_corr:

CX\_corr=[1.0000 0.9940 -0.9904 ; 0.9940 1.0000 -0.9782 ; -0.9904 -0.9782 1.0000];

X=unc([x1 x2 x3] , CX\_corr ,'corr');

In order to transform the correlation matrix CX\_corr into its corresponding covariance matrix CX\_corr, the method cor\_into\_cov\_mat(X,CX\_corr) is used, where:

X: array of uncertainty objects

CX\_corr: the correlation matrix of the elements of X

Therefore, using the cor\_into\_cov\_mat() for this example yields:

CX\_cov = cor\_into\_cov\_mat([x1 x2 x3],CX\_corr);

% CX\_cov =

%

% 13.7663 6.8996 -16.8661

% 6.8996 3.4999 -8.3994

% -16.8661 -8.3994 21.0663

**Note:** The methods in sections 8.3 and 8.4 are not implemented in MC and UT approaches.

# Uncertainty Propagation in Real-Valued Quantities

## Addition of Real-Valued Quantities

### Example 1

Consider two objects a and b:

a=unc(3.5,0.1);

b=unc(5.7,0.3);

% a =

%

% 3.50(10)

%

% b =

%

% 5.70(30)

The sum s of a and b is:

s = a + b;

% s =

%

% 9.20(32)

Check the mean value of s:

gmv(s);

% ans =

%

% 9.2000

Check the standard uncertainty of s:

gmu(s);

% ans =

%

% 0.3162

### Example 2

Consider two objects a and b:

a=unc(3.5,0.1);

b=unc(5.7,0.3);

% a =

%

% 3.50(10)

%

% b =

%

% 5.70(30)

Assign names to a and b:

a.name='a';

b.name='b';

Set the correlation coefficient between a and b to 0.5:

set\_correl(a,b,0.5);

The sum s of a and b is:

s = a + b;

% s =

%

% 9.20(36)

Check the mean value of s:

gmv(s);

% ans =

%

% 9.2000

Check the standard uncertainty of s:

gmu(s);

% ans =

%

% 0.3606

**Note:** The name property and the method set\_correl are not available in MC and UT approaches

## Addition of a Real-Valued Quantity and a Constant Number

### Example

Consider an object *a* and a constant number *c*:

a = unc( 13 , 0.5 );

c = 2.1;

The sum s of a and c is:

s = a + 2.1;

% s =

%

% 15.10(50)

## Subtraction of Real-Valued Quantities

### Example 1

Consider two objects a and b:

a=unc(6.4,0.2);

b=unc(5.3,0.1);

% a =

%

% 6.40(20)

%

% b =

%

% 5.30(10)

The subtraction of b from a is:

c = a - b;

% c =

%

% 1.10(22)

Check the mean value of c:

gmv(c);

% ans =

%

% 1.1000

Check the standard uncertainty of c:

gmu(c);

% ans =

%

% 0.2236

### Example 2

Consider two objects a and b:

a=unc(6.4,0.2);

b=unc(5.3,0.1);

% a =

%

% 6.40(20)

%

% b =

%

% 5.30(10)

Assign names to a and b:

a.name='a';

b.name='b';

Set the covariance between a and b to 0.01:

set\_cov(a,b,0.01);

The subtraction of b from a is:

c = a - b;

% c =

%

% 1.10(17)

Check the mean value of c:

gmv(c);

% ans =

%

% 1.1000

Check the standard uncertainty of c:

gmu(c);

% ans =

%

% 0.1732

**Note:** The name property and the method set\_cov are not available in MC and UT approaches

## Subtraction of a Real-Valued Quantity and a Constant Number

### Example

Consider an object *a* and a constant number *c*:

a = unc( 13 , 0.5 );

c = 2.1;

The subtraction of c from a is:

b = a - c;

% b =

%

% 10.90(50)

## Product of Real-Valued Quantities

### Example 1

Consider two objects a and b:

a=unc(2.0,0.3);

b=unc(3.0,0.3);

% a =

%

% 2.00(30)

%

% b =

%

% 3.00(30)

The product p of a and b is:

p = a \* b;

% p =

%

% 6.0(1.1)

Check the mean value of p:

gmv(p);

% ans =

%

% 6

Check the standard uncertainty of p:

gmu(p);

% ans =

%

% 1.0817

### Example 2

Consider two objects a and b:

a=unc(2.0,0.3);

b=unc(3.0,0.3);

% a =

%

% 2.00(30)

%

% b =

%

% 3.00(30)

Assign names to a and b:

a.name='a';

b.name='b';

Set the correlation coefficient between a and b to 0.5:

set\_correl(a,b,0.5);

The product p of a and b is:

p = a \* b;

% p =

%

% 6.0(1.3)

Check the mean value of p:

gmv(p);

% ans =

%

% 6

Check the standard uncertainty of p:

gmu(p);

% ans =

%

% 1.3077

**Note:** The name property and the method set\_correl are not available in MC and UT approaches

## Product of a Real-Valued Quantity and a Constant Number

**Example:**

Consider an object *a* and a constant number *c*:

a = unc( 8 , 0.3 );

c = 34;

The product p of a and c is:

p = a \* c;

% p =

%

% 272(10)

## Ratio of Real-Valued Quantities

### Example 1

Consider two objects a and b:

a=unc(5.0,0.1);

b=unc(2.0,0.2);

% a =

%

% 5.00(10)

%

% b =

%

% 2.00(20)

The ratio q of a to b is:

q = a / b;

% q =

%

% 2.50(25)

Check the mean value of q:

gmv(q);

% ans =

%

% 2.5000

Check the standard uncertainty of q:

gmu(q);

% ans =

%

% 0.2550

### Example 2

Consider two objects a and b:

a=unc(5.0,0.1);

b=unc(2.0,0.2);

% a =

%

% 5.00(10)

%

% b =

%

% 2.00(20)

Assign names to a and b:

a.name='a';

b.name='b';

Set the correlation coefficient between a and b to 0.5:

set\_correl(a,b,0.5);

The ratio q of a to b is:

q = a / b;

% q =

%

% 2.50(23)

Check the mean value of q:

gmv(q);

% ans =

%

% 2.5000

Check the standard uncertainty of q:

gmu(q);

% ans =

%

% 0.2291

**Note:** The name property and the method set\_correl are not available in MC and UT approaches

## Ratio of a Real-Valued Quantity and a Constant Number

### Example 1

Consider an object *a* and a constant number *c*:

a = unc( 21 , 0.2 );

c = 13;

The ratio q of a to c is:

q = a / c;

% q =

%

% 1.615(15)

### Example 2

Consider an object *a* and a constant number *c*:

a = unc( 21 , 0.2 );

c = 13;

The ratio q of c to a is:

q = c / a;

% q =

%

% 0.6190(59)

## Real Logarithm Functions

### Real Natural Logarithm Function log()

**Example:**

a = unc( 21 , 0.2 );

log(a);

% ans =

%

% 3.0445(95)

### Real Decimal Logarithm Function log10()

**Example:**

a = unc( 21 , 0.2 );

log10(a);

% ans =

%

% 1.3222(41)

## Real Exponential Function exp()

## Example:

a = unc( 3.4 , 0.5 );

b = exp(a);

% b =

%

% 30(15)

Check the mean value of b:

gmv(b)

% ans =

%

% 29.9641

Check the standard uncertainty of b:

gmu(b)

% ans =

%

% 14.9821

## Real Trigonometric Functions

### Real Sine Function sin()

**Example:**

x = unc( 3.5 , 0.1);

sin(x);

% ans =

%

% -0.351(94)

### Real Cosine Function cos()

**Example:**

x = unc( 3.5 , 0.1);

cos(x);

% ans =

%

% -0.936(35)

### Real Tangent Function tan()

**Example:**

x = unc( 3.5 , 0.1);

tan(x);

% ans =

%

% 0.37(11)

### Real Cotangent Function cot()

**Example:**

x = unc( 3.5 , 0.1);

cot(x);

% ans =

%

% 2.67(81)

### Inverse Sine Function asin()

**Example:**

y = unc( -0.3508 , 0.0936);

asin(y)

% ans =

%

% -0.358(100)

### Inverse Cosine Function acos()

**Example:**

y = unc( -0.9365 , 0.0351);

acos(y)

% ans =

%

% 2.78(10)

### Inverse Tangent Function atan()

**Example:**

y = unc( 0.3746 , 0.1140 );

atan(y)

% ans =

%

% 0.358(100)

### Inverse Cotangent Function acot()

**Example:**

y = unc( 2.6696 , 0.8127 );

acot(y)

% ans =

%

% 0.36(10)

### Four-Quadrant Inverse Tangent Function atan2()

**Example:**

x1=unc(7.8333, 3.7103);

x2=unc(3.5000, 1.8708);

atan2(x1,x2)

% ans =

%

% 1.15(27)

## Real Hyperbolic Functions

### Real Hyperbolic Sine Function sinh()

**Example:**

x = unc( 3.5 , 0.1);

y=sinh(x);

% y =

%

% 16.5(1.7)

### Real Hyperbolic Cosine Function cosh()

**Example:**

x = unc( 3.5 , 0.1);

y=cosh(x);

% y =

%

% 16.6(1.7)

### Real Hyperbolic Tangent Function tanh()

**Example:**

x = unc( 3.5 , 0.1);

y=tanh(x);

% y =

%

% 0.99818(36)

### Real Hyperbolic Cotangent Function coth()

**Example:**

x = unc( 3.5 , 0.1);

y=coth(x);

% y =

%

% 1.00183(37)

### Inverse Hyperbolic Sine Function asinh()

**Example:**

y = unc( 16.5426 , 1.6573 );

asinh(y)

% ans =

%

% 3.50(10)

### Inverse Hyperbolic Cosine Function acosh()

**Example:**

y = unc( 16.5728 , 1.6543 );

acosh(y)

% ans =

%

% 3.50(10)

### Inverse Hyperbolic Tangent Function atanh()

**Example:**

y = unc( 0.9982 , 3.6409e-04 );

atanh(y)

% ans =

%

% 3.51(10)

## Square Root Function sqrt()

**Example:**

x = unc( 16.5728 , 1.6543 );

sqrt(x)

% ans =

%

% 4.07(20

## Covariance and Correlation Propagation

### Example 1

Consider two real objects x1 and x2:

x1=unc(0.8700,0.7103,'x1');

x2=unc(0.9400,0.5337,'x2');

Enter the corresponding covariance matrix CX:

CX= [ 0.5046 -0.0864 ; -0.0864 0.2849 ];

unc([x1 x2] , CX ,'cov');

Define two output objects y1 and y2:

y1= x1 + x2;

y2= x1 - x2;

Find the covariance between y1 and y2 using the method *covr():*

[cov,CY] = covr( y1 , y2 );

% cov =

%

% 0.2197

%

%

% CY =

%

% 0.6166 0.2197

% 0.2197 0.9622

CY is the output covariance matrix of the measurement model defined by :

y1= x1 + x2;

y2= x1 - x2;

**Note**: Syntax 3 and function covr are not implemented in MC and UT approaches

### Example 2

Consider three objects x1, x2, and x3:

x1=unc(1.8100,0.7852,'x1');

x2=unc(1.9100,0.8465,'x2');

x3=unc(2.2900,0.8569,'x3');

Enter the corresponding covariance matrix CX:

CX = [ 0.6166 0.6154 0.6057 ; 0.6154 0.7166 0.6934 ; 0.6057 0.6934 0.7343];

unc([x1 x2 x3] , CX ,'cov');

Define two output objects y1 and y2:

y1= 2\*x1 + 3\*x2;

y2= x1 - x3;

Find the covariance between y1 and y2 using the method *covr():*

[cov,CY] = covr( y1 , y2 );

% cov =

%

% -0.2123

%

%

% CY =

%

% 16.3000 -0.2123

% -0.2123 0.1394

CY is the output covariance matrix of the measurement model defined by :

y1= 2\*x1 + 3\*x2;

y2= x1 - x3;

### Example 3

Consider three objects x1, x2, and x3:

x1=unc(0.8700, 0.7103 , 'x1' );

x2=unc(0.9400, 0.5337 , 'x2' );

x3=unc(2.1130, 0.5759 , 'x3' );

Enter the corresponding covariance matrix CX:

CX=[ 0.5046 -0.0864 -0.1562 ; -0.0864 0.2849 0.0164 ; -0.1562 0.0164 0.3317];

unc([x1 x2 x3] , CX ,'cov');

Define two output objects y1 and y2:

y1=x1 + x2 ;

y2=x3-x1;

Find the covariance between y1 and y2 using the method *covr():*

[cov,CY] = covr( y1 , y2);

% cov =

%

% -0.5579

%

%

% CY =

%

% 0.6166 -0.5579

% -0.5579 1.1486

Define two other output objects z1 and z2:

z1= y1 + y2;

z2= y1 - y2;

Find the covariance between z1 and z2 using the method *covr():*

[cov,CZ] = covr( z1 , z2);

% cov =

%

% -0.5320

%

%

% CZ =

%

% 0.6493 -0.5320

% -0.5320 2.8810

CZ is the output covariance matrix of the measurement model defined by:

y1=x1 + x2 ;

y2=x3-x1;

z1= y1 + y2;

z2= y1 - y2;

**Note:** Syntax 3 and function covr are not available in MC and UT approaches

## Karhunen-Loeve Decomposition

### Example 1

Consider two real objects x1 and x2:

x1=unc(7.8333, 3.7103 , 'x1');

x2=unc(3.5000, 1.8708 , 'x2');

The corresponding covariance matrix CX is given by:

CX=[13.7667 6.9000; 6.9000 3.5000];

Decorrelate x1 and x2:

[K, Coef\_mat]=karhloev( [x1 x2],CX,'cov' );

% K =

%

% 0.39(18)

% -8.6(4.2)

%

% Coef\_mat =

%

% 0.4489 -0.8936

% -0.8936 -0.4489

Check if the decorrelation is correctly performed:

X = Coef\_mat \* K;

% X =

%

% 7.8(3.7)

% 3.5(1.9)

The real object x1 can be replaced by a new real object y1 given by:

Coef\_mat(1,:) \* K;

% ans =

%

% 7.8(3.7)

The real object x2 can be replaced by a new real object y2 given by:

Coef\_mat(2,:) \* K ;

% ans =

%

% 3.5(1.9)

The new real objects y1 and y2 are uncorrelated.

**Note:** The method karhloev is not yet implemented in MC and UT approaches

### Example 2

Consider two real objects x1 and x2:

x1=unc(7.8333, 3.7103 , 'x1');

x2=unc(3.5000, 1.8708 , 'x2');

The corresponding correlation matrix CX is given by:

CX = [ 1.0000 0.9941 ; 0.9941 1.0000];

Decorrelate x1 and x2:

[K, Coef\_mat]=karhloev( [x1 x2],CX,'corr' )

% K =

%

% 0.39(18)

% -8.6(4.2)

%

% Coef\_mat =

%

% 0.4490 -0.8936

% -0.8936 -0.4490

Check if the decorrelation is correctly performed:

X = Coef\_mat \* K;

% X =

%

% 7.8(3.7)

% 3.5(1.9)

The real object x1 can be replaced by a new real object y1 given by:

Coef\_mat(1,:) \* K;

% ans =

%

% 7.8(3.7)

The real object x2 can be replaced by a new real object y2 given by:

Coef\_mat(2,:) \* K ;

% ans =

%

% 3.5(1.9)

The new real objects y1 and y2 are uncorrelated.

**Note:** The method karhloev is not yet implemented in MC and UT approaches

### Example 3

Consider two real objects x1 and x2:

x1=unc(7.8333, 3.7103 , 'x1');

x2=unc(3.5000, 1.8708 , 'x2');

Set the covariance between x1 and x2 to 6.9:

set\_cov(x1,x2,6.9);

Decorrelate x1 and x2:

[K, Coef\_mat]=karhloev([x1 x2]);

% K =

%

% 0.39(18)

% -8.6(4.2)

%

% Coef\_mat =

%

% 0.4490 -0.8936

% -0.8936 -0.4490

Check if the decorrelation is correctly performed:

X = Coef\_mat \* K;

% X =

%

% 7.8(3.7)

% 3.5(1.9)

The real object x1 can be replaced by a new real object y1 given by:

Coef\_mat(1,:) \* K;

% ans =

%

% 7.8(3.7)

The real object x2 can be replaced by a new real object y2 given by:

Coef\_mat(2,:) \* K;

% ans =

%

% 3.5(1.9)

The new real objects y1 and y2 are uncorrelated.

**Note:** The method karhloev is not yet implemented in MC and UT approaches

# Uncertainty Propagation in Complex-Valued Quantities

## Creation of Complex Uncertainty Objects

Complex uncertainty objects can be created in different ways as needed.

### Example 1

Create real uncertainty objects

a=unc(1,0.1);

b=unc(2,0.2);

Create the complex uncertainty object *Z* using the overloaded function complex()

Z=complex(a,b);

% Z =

%

% 1.00(10) + 2.00(20) \* i

To check the real and imaginary parts:

real(Z)

% ans =

%

% 1.00(10)

imag(Z)

% ans =

%

% 2.00(20)

### Example 2

Create real uncertainty objects:

a=unc(1,0.1);

b=unc(2,0.2);

Create the complex uncertainty object *Z :*

Z = a + 1i \* b ;

% Z =

%

% 1.00(10) + 2.00(20) \* i

### Example 3

The complex uncertainty object can be directly created:

Z = unc(1,0.1) + 1i \* unc(2,0.2) ;

% Z =

%

% 1.00(10) + 2.00(20) \* i

### Example 4

Arrays of complex objects can be created.

Consider the following real objects:

a=unc(1,0.1);

b=unc(2,0.2);

c=unc(3,0.3);

d=unc(5,0.4);

e=unc(8,0.5);

f=unc(13,0.6);

g=unc(21,0.7);

h=unc(34,0.8);

Create arrays X and Y:

X=[ a b; c d];

Y=[e f; g h];

Create the array Z of complex objects out of X and Y:

Z= complex(X,Y);

% Z =

%

% 1.00(10) + 8.00(50) \* i 2.00(20) + 13.00(60) \* i

% 3.00(30) + 21.00(70) \* i 5.00(40) + 34.00(80) \* i

## Addition of Complex Uncertainty Objects

### Example

Consider two complex objects z1 and z2:

z1=complex(unc(65.4,2.2),unc(-6.2,0.4));

z2=complex(unc(7.6,0.9),unc(53.2,3.4));

The sum of z1 and z2 is:

z=z1 + z2;

% z =

%

% 73.0(2.4) + 47.0(3.4) \* i

## Subtraction of Complex Uncertainty Objects

### Example

Consider two complex objects z1 and z2:

z1=complex(unc(65.4,2.2),unc(-6.2,0.4));

z2=complex(unc(7.6,0.9),unc(53.2,3.4));

The subtraction of z1 from z2 is:

z=z2-z1;

% z =

%

% -57.8(2.4) + 59.4(3.4) \* i

## Product of Complex Uncertainty Objects

### Example

Consider two complex objects z1 and z2:

z1=complex(unc(32.7,1.1),unc(-3.1,0.2));

z2=complex(unc(7.6,0.9),unc(53.2,3.4));

The product of z1 and z2 is:

Z = z1 \* z2;

% Z =

%

% 413(34) + 1720(130) \* i

Check the properties of the real part of Z:

Z\_real = real(Z) ;

gmv(Z\_real)

% ans =

%

% 413.4400

gmu(Z\_real)

% ans =

%

% 34.0634

Check the properties of the imaginary part of Z:

Z\_imag = imag(Z) ;

gmv(Z\_imag)

% ans =

%

% 1.7161e+03

gmu(Z\_imag)

% ans =

%

% 125.6809

## Mixed Product of Complex and Real Uncertainty Objects

### Example

Consider the real object a:

a=unc(2.7,0.2);

Consider the complex object z:

z=complex(unc(32.7,1.1),unc(-3.1,0.2));

The mixed product w of a and z is:

w = a \* z ;

% w =

%

% 88.3(7.2) - 8.37(82) \* i

## Ratio of Complex Uncertainty Objects

### Example

Consider two complex objects z1 and z2:

z1=complex(unc(32.7,1.1),unc(-3.1,0.2));

z2=complex(unc(7.6,0.9),unc(53.2,3.4));

The ratio w of z1 to z2 is:

w = z1 / z2;

% w =

%

% 0.029(13) - 0.611(43) \* i

Check the properties of the real part of w:

w\_real = real(w) ;

gmv(w\_real)

% ans =

%

% 0.0289

gmu(w\_real)

% ans =

%

% 0.0133

Check the properties of the imaginary part of w:

w\_imag = imag(w);

gmv(w\_imag)

% ans =

%

% -0.6105

gmu(w\_imag)

% ans =

%

% 0.0431

## Mixed Ratio of Complex and Real Uncertainty Objects

### Example 1

Consider the real object a:

a=unc(2.3,0.1);

Consider the complex object z:

z=complex(unc(7.6,0.9),unc(53.2,3.4));

The mixed ratio w of z to a is:

w = z / a;

% w =

%

% 3.30(42) + 23.1(1.8) \* i

### Example 2

Consider the real object a:

a=unc(2.3,0.1);

Consider the complex object z:

z=complex(unc(7.6,0.9),unc(53.2,3.4));

The mixed ratio w of a to z is:

w = a / z;

% w =

%

% 0.0061(11) - 0.0424(32) \* i

## Absolute Value of a Complex Object

### Example

Consider the complex object z:

z=complex(unc(32.7,1.1),unc(-3.1,0.2));

The absolute value of z is:

abs(z)

% ans =

%

% 32.8(1.1)

## Conjugate of a Complex Object

### Example

Consider the complex object z:

z=complex(unc(32.7,1.1),unc(-3.1,0.2));

The conjugate of z is:

conj(z)

% ans =

%

% 32.7(1.1) + 3.10(20) \* i

## Complex Functions

### Complex Logarithmic Functions

### Complex Natural Logarithm Function log()

### Example:

Consider the complex object z:

z = complex(unc(7.6,0.9),unc(53.2,3.4));

The natural logarithm of z is:

w=log(z);

% w =

%

% 3.984(63) + 1.429(19) \* i

%

Check the properties of the real part of w:

w\_real = real(w);

gmv(w\_real)

% ans =

%

% 3.9842

%

gmu(w\_real)

% ans =

%

% 0.0627

Check the properties of the imaginary part of w:

w\_imag = imag(w);

gmv(w\_imag)

% ans =

%

% 1.4289

%

gmu(w\_imag)

% ans =

%

% 0.0188

### Complex Decimal Logarithm Function log10()

### Example:

Consider the complex object z:

z = complex(unc(7.6,0.9),unc(53.2,3.4));

The decimal logarithm of z is:

w = log10(z);

% w =

%

% 1.730(27) + 0.6206(82) \* i

### Complex Exponential Function exp()

### Example:

Consider the complex object z:

z = complex(unc(7.6,0.9),unc(53.2,3.4));

The exponential of z is:

w=exp(z);

% w =

%

% -2000(2200) + 400(6700) \* i

Check the properties of the real part of w:

w\_real = real(w) ;

gmv(w\_real)

% ans =

%

% -1.9555e+03

gmu(w\_real)

% ans =

%

% 2.2469e+03

Check the properties of the imaginary part of w:

w\_imag = imag(w) ;

gmv(w\_imag)

% ans =

%

% 410.8258

gmu(w\_imag)

% ans =

%

% 6.6590e+03

### Complex Trigonometric Functions

### Complex Cosine Function cos()

### Example:

Consider the complex object z:

z = complex(unc(7.6,0.9),unc(53.2,3.4));

The cosine of z is:

w=cos(z);

% w =

%

% 1.6e+22(7.8e+22) - 6e+22(2.1e+23) \* i

%

Check the properties of the real part of w:

w\_real = real(w);

gmv(w\_real)

% ans =

%

% 1.5979e+22

gmu(w\_real)

% ans =

%

% 7.7595e+22

Check the properties of the imaginary part of w:

w\_imag = imag(w) ;

gmv(w\_imag)

% ans =

%

% -6.1557e+22

gmu(w\_imag)

% ans =

%

% 2.0979e+23

### Complex Sine Function sin()

### Example:

Consider the complex object z:

z = complex(unc(7.6,0.9),unc(53.2,3.4))

The sine of z is:

w=sin(z);

% w =

%

% 6e+22(2.1e+23) + 1.6e+22(7.8e+22) \* i

Check the properties of the real part of w:

w\_real = real(w) ;

gmv(w\_real)

% ans =

%

% 6.1557e+22

gmu(w\_real)

% ans =

%

% 2.0979e+23

Check the properties of the imaginary part of w:

w\_imag = imag(w) ;

gmv(w\_imag)

% ans =

%

% 1.5979e+22

gmu(w\_imag)

% ans =

%

% 7.7595e+22

### Complex Tangent Function tan()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The tangent of z is:

w = tan(z);

% w =

%

% 0.75(27) + 1.71(51) \* i

### Complex Cotangent Function cot()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The cotangent of z is:

w = cot(z);

% w =

%

% 0.21(16) - 0.490(43) \* i

### Complex Inverse Sine Function asin()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The inverse sine of z is:

w = asin(z);

% w =

%

% 1.09(34) + 0.93(26) \* i

### Complex Inverse Cosine Function acos()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The inverse cosine of z is:

w = acos(z);

% w =

%

% 0.484(13) - 0.93(16) \* i

### Complex Inverse Tangent Function atan()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The inverse tangent of z is:

w = atan(z);

% w =

%

% 0.959(64) + 0.177(34) \* i

### Complex Inverse Cotangent Function acot()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The inverse cotangent of z is:

w = acot(z);

% w =

%

% 0.612(64) - 0.177(34) \* i

### Complex Hyperbolic Functions

### Complex Hyperbolic Sine Function sinh()

### Example:

Consider the complex object z:

z = complex(unc(7.6,0.9),unc(53.2,3.4));

The hyperbolic sine of z is:

w=sinh(z);

% w =

%

% -1000(1100) - 200(3300) \* i

Check the properties of the real part of w:

w\_real = real(w) ;

gmv(w\_real)

% ans =

%

% -977.7534

gmu(w\_real)

% ans =

%

% 1.1234e+03

Check the properties of the imaginary part of w:

w\_imag = imag(w) ;

gmv(w\_imag)

% ans =

%

% 205.4130

gmu(w\_imag)

% ans =

%

% 3.3295e+03

### Complex Hyperbolic Cosine Function cosh()

### Example:

Consider the complex object z:

z = complex(unc(7.6,0.9),unc(53.2,3.4));

The hyperbolic cosine of z is:

w=cosh(z);

% w =

%

% -1000(1100) + 200(3300) \* i

Check the properties of the real part of w:

w\_real = real(w) ;

gmv(w\_real)

% ans =

%

% -977.7539

gmu(w\_real)

% ans =

%

% 1.1234e+03

Check the properties of the imaginary part of w:

w\_imag = imag(w) ;

gmv(w\_imag)

% ans =

%

% 205.4129

gmu(w\_imag)

% ans =

%

% 3.3295e+03

### Complex Hyperbolic Tangent Function tanh()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The hyperbolic tangent of z is:

w = tanh(z);

% w =

%

% 0.916(35) + 0.115(42) \* i

### Complex Hyperbolic Cotangent Function coth()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The hyperbolic cotangent of z is:

w = coth(z);

% w =

%

% 1.075(28) - 0.135(58) \* i

### Complex Inverse Hyperbolic Sine Function asinh()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The inverse hyperbolic sine of z is:

w = asinh(z);

% w =

%

% 1.12(12) + 0.301(16) \* i

### Complex Inverse Hyperbolic Cosine Function acosh()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The inverse hyperbolic cosine of z is:

w = acosh(z);

% w =

%

% 0.93(12) + 0.484(43) \* i

### Complex Inverse Hyperbolic Tangent Function atanh()

### Example:

Consider the complex object z:

z = complex(unc(1.3,0.2),unc(0.5,0.01));

The inverse hyperbolic tangent of z is:

w = atanh(z);

% w =

%

% 0.698(47) + 1.16(14) \* i

## Complex Square Root Function sqrt()

**Example:**

z = complex(unc(32.7,1.1),unc(-3.1,0.2)) ;

% z =

%

% 32.7(1.1) - 3.10(20) \* i

sqrt(z)

% ans =

%

% 5.725(95) - 0.271(22) \* i

## Covariance and Correlation Calculation

### Covariance and Correlation between Real and Imaginary Parts of a Complex-Valued Quantity

### Example:

Consider two real objects a and b:

a=unc(7.8333, 3.7103 , 'a');

b=unc(3.5000, 1.8708 , 'b');

Set covariance between a and b to 0.6:

set\_cov(a,b,0.6);

In order to be able to work with correlations and covariances, the complex object must be generated using the following syntax:

z=complex(a,b);

% z =

%

% 7.8(3.7) + 3.5(1.9) \* i

The covariance between the real and imaginary parts of z can then be checked:

covr(z.r , z.img)

% ans =

%

% 0.6000

The correlation between the real and imaginary parts of z can also be checked:

correl(z.r , z.img)

% ans =

%

% 0.0864

If the exponential of z is taken in this case where the real and imaginary parts of z are correlated, then the result is:

exp(z)

% ans =

%

% -2400(8800) - 900(5700) \* i

If the covariance of the real and imaginary parts of z are set to zero:

set\_cov(z.r , z.img , 0);

Then the result of taking the exponential of z is:

exp(z)

% ans =

%

% -2400(8900) - 900(5500) \* i

**Note:** The methods set\_cov, set\_correl and the properties r, img are not available in MC and UT approaches

### Covariance and Correlation between Real and Imaginary Parts of more than One Complex-Valued Quantity

**Example:**

Consider three real objects a,b, and c:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

c=unc(2.2900,0.8569,'c');

Enter the corresponding covariance matrix CX:

CX = [ 0.6166 0.6154 0.6057 ; 0.6154 0.7166 0.6934 ; 0.6057 0.6934 0.7343];

unc([a b c] , CX ,'cov');

Create two complex objects z1 and z2:

z1=complex(a,b);

z2=complex(b,c);

% z1 =

%

% 1.81(79) + 1.91(85) \* i

%

% z2 =

%

% 1.91(85) + 2.29(86) \* i

Check the covariance between the real part of z1 and the imaginary part of z2:

covr(z1.r,z2.img);

% ans =

%

% 0.6057

Check the covariance between the imaginary part of z1 and the real part of z2:

covr(z1.img,z2.r);

% ans =

%

% 0.7166

Perform the product of z1 and z2:

z1 \* z2 ;

% ans =

%

% -0.92(90) + 7.8(6.4) \* i

If the covariances between a,b, and c are set to zero:

set\_cov(a,b,0);

set\_cov(a,c,0);

set\_cov(c,b,0);

Then the result of the product of z1 and z2 becomes:

z1 \* z2 ;

% ans =

%

% -0.9(2.3) + 7.8(4.0) \* i

**Note:** The methods set\_cov, set\_correl, covr and the properties r, img are not available in MC and UT approaches

### Covariance and Correlation between Two Complex-Valued Quantities

**Example:**

Consider three real objects a,b, and c:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

c=unc(2.2900,0.8569,'c');

Enter the corresponding covariance matrix CX:

CX = [ 0.6166 0.6154 0.6057 ; 0.6154 0.7166 0.6934 ; 0.6057 0.6934 0.7343];

unc([a b c] , CX ,'cov');

Create two complex objects z1 and z2:

z1=complex(a,b);

z2=complex(b,c);

% z1 =

%

% 1.81(79) + 1.91(85) \* i

%

% z2 =

%

% 1.91(85) + 2.29(86) \* i

Find the covariance between z1 and z2:

covr(z1,z2);

% ans =

%

% 0.6165 0.6154 0.6154 0.6057

% 0.6154 0.7166 0.7166 0.6934

% 0.6154 0.7166 0.7166 0.6934

% 0.6057 0.6934 0.6934 0.7343

Find the correlation between z1 and z2:

correl(z1,z2);

% ans =

%

% 1.0000 0.9258 0.9258 0.9002

% 0.9258 1.0000 1.0000 0.9559

% 0.9258 1.0000 1.0000 0.9559

% 0.9002 0.9559 0.9559 1.0000

**Note:** The methods set\_cov, set\_correl,covr,correl and syntax 3 are not available in MC and UT approaches

### Covariance and Correlation between a Complex-Valued Quantity and a Real Object

**Example:**

Consider three real objects a,b, and c:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

c=unc(2.2900,0.8569,'c');

Enter the corresponding covariance matrix CX:

CX = [ 0.6166 0.6154 0.6057 ; 0.6154 0.7166 0.6934 ; 0.6057 0.6934 0.7343];

unc([a b c] , CX ,'cov');

Create the complex object z:

z = complex(a,b);

% z =

%

% 1.81(79) + 1.91(85) \* i

Check the covariance between z and c:

covr(z,c);

% ans =

%

% 0.6057 0.6934

Check the correlation between z and c:

correl(z,c)

% ans =

%

% 0.9002 0.9559

**Note:** The methods set\_cov, set\_correl, covr, correl and syntax 3 are not available in MC and UT approaches

## Karhunen-Loeve Decomposition

### Karhunen-Loeve Decomposition of One Complex Object

**Example:**

Consider two real objects a and b:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

Set covariance between a and b to 0.6:

set\_cov(a,b,0.6);

Create the complex object z:

z=complex(a,b);

% z =

%

% 1.81(79) + 1.91(85) \* i

Decorrelate z :

[K, Coef\_mat]=karhloev(z);

% K =

%

% -0.04(25)

% 2.6(1.1)

%

% Coef\_mat =

%

% -0.7359 0.6771

% 0.6771 0.7359

### Karhunen-Loeve Decomposition of more than One Complex Object

**Example:**

Consider two real objects a and b:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

Set covariance between a and b to 0.6:

set\_cov(a,b,0.6);

Create two complex objects z1 and z2:

z1=complex(a,b);

z2=complex(b,a);

% z1 =

%

% 1.81(79) + 1.91(85) \* i

%

% z2 =

%

% 1.91(85) + 1.81(79) \* i

Create an array Z:

Z=[z1 z2];

% Z =

%

% 1.81(79) + 1.91(85) \* i 1.91(85) + 1.81(79) \* i

Perform decorrelation of z1 and z2:

[K, Coef\_mat]=karhloev(Z);

% K =

%

% -0.04(25)

% 2.6(1.1)

%

% Coef\_mat =

%

% -0.7359 0.6771

% 0.6771 0.7359

### Karhunen-Loeve Decomposition of Mixed Complex and Real Objects

**Example:**

Consider three real objects a,b, and c:

a=unc(1.8100,0.7852,'a');

b=unc(1.9100,0.8465,'b');

c=unc(2.2900,0.8569,'c');

Enter the corresponding covariance matrix CX:

CX = [ 0.6166 0.6154 0.6057 ; 0.6154 0.7166 0.6934 ; 0.6057 0.6934 0.7343];

unc([a b c] , CX ,'cov');

Create the complex object z:

z=complex(a,c);

% z =

%

% 1.81(79) + 2.29(86) \* i

Decorrelate a,b, and z:

[K, Coef\_mat]=karhloev([ a z b]);

% K =

%

% -0.23(17)

% -0.14(26)

% 3.5(1.4)

%

% Coef\_mat =

%

% -0.2323 0.8104 0.5379

% 0.7859 -0.1695 0.5947

% -0.5731 -0.5608 0.5975

Now, the real object *a* can be replaced by another real object *a1* given by:

Coef\_mat(1,:) \* K;

% ans =

%

% 1.81(79)

The complex object z can be replaced by another complex object *z1* with a real part given by:

Coef\_mat(1,:) \* K;

% ans =

%

% 1.81(79)

( because the real part of z is a ) and with an imaginary part given by:

Coef\_mat(3,:) \* K;

% ans =

%

% 2.29(86)

( because the imaginary part of z is c ).

The real object *b* can be replaced by another real object *b1* given by:

Coef\_mat(2,:) \* K;

% ans =

%

% 1.91(85)

The new uncertainty objects a1, z1, and b1 are uncorrelated.

**Note:** The method karhloev is not implemented in MC and UT approaches

# Special Operations on Arrays

## Determinant

### Example

Consider four objects x1, x2, x3, and x4:

x1=unc(10,1,'x1');

x2=unc(20,2,'x2');

x3=unc(30,3,'x3');

x4=unc(40,4,'x4');

Create the array A:

A=[ x1 x2; x3 x4 ];

% A =

%

% 10.0(1.0) 20.0(2.0)

% 30.0(3.0) 40.0(4.0)

The determinant of A is:

det(A);

% ans =

%

% -200(100)

## Inverse Matrix

### Example

Consider four objects x1, x2, x3, and x4:

x1=unc(10,1,'x1');

x2=unc(20,2,'x2');

x3=unc(30,3,'x3');

x4=unc(40,4,'x4');

Create the array A:

A=[ x1 x2; x3 x4 ];

% A =

%

% 10.0(1.0) 20.0(2.0)

% 30.0(3.0) 40.0(4.0)

The inverse matrix of A is:

inv(A);

% ans =

%

% -0.20(11) 0.100(46)

% 0.150(69) -0.050(28)

## Diagonal Elements of a Matrix

### Example

Consider four objects x1, x2, x3, and x4:

x1=unc(10,1,'x1');

x2=unc(20,2,'x2');

x3=unc(30,3,'x3');

x4=unc(40,4,'x4');

Create the array A:

A=[ x1 x2; x3 x4 ];

% A =

%

% 10.0(1.0) 20.0(2.0)

% 30.0(3.0) 40.0(4.0)

The diagonal elements of the matrix of A are found using the command:

diag(A);

% ans =

%

% 10.0(1.0) 40.0(4.0)

## Trace of a Matrix

### Example

Consider four objects x1, x2, x3, and x4:

x1=unc(10,1,'x1');

x2=unc(20,2,'x2');

x3=unc(30,3,'x3');

x4=unc(40,4,'x4');

Create the array A:

A=[ x1 x2; x3 x4 ];

% A =

%

% 10.0(1.0) 20.0(2.0)

% 30.0(3.0) 40.0(4.0)

The trace of the matrix of A is:

trace(A);

% ans =

%

% 50.0(4.1)

## Largest Element in a 1xn or nx1 Array

### Example

Consider four objects x1, x2, x3, and x4:

x1=unc(10,1,'x1');

x2=unc(20,2,'x2');

x3=unc(30,3,'x3');

x4=unc(40,4,'x4');

Create the 1x4 array A:

A=[ x1 x2 x3 x4 ];

% A =

%

% 10.0(1.0) 20.0(2.0) 30.0(3.0) 40.0(4.0)

The largest element of the 1x4 array A is:

max(A);

% ans =

%

% 40.0(4.0)

## Smallest Element in a 1xn or nx1 Array

### Example

Consider four objects x1, x2, x3, and x4:

x1=unc(10,1,'x1');

x2=unc(20,2,'x2');

x3=unc(30,3,'x3');

x4=unc(40,4,'x4');

Create the 1x4 array A:

A=[ x1 x2 x3 x4 ];

% A =

%

% 10.0(1.0) 20.0(2.0) 30.0(3.0) 40.0(4.0)

The smallest element of the 1x4 array A is:

min(A);

% ans =

%

% 10.0(1.0)

## Sum of Array Elements

### Example

Consider four objects x1, x2, x3, and x4:

x1=unc(10,1,'x1');

x2=unc(20,2,'x2');

x3=unc(30,3,'x3');

x4=unc(40,4,'x4');

Create the array A:

A=[ x1 x2; x3 x4 ];

% A =

%

% 10.0(1.0) 20.0(2.0)

% 30.0(3.0) 40.0(4.0)

The sum of the elements of the array A is:

sum(A);

% ans =

%

% 100.0(5.5)

## Cumulative Sum of Array Elements

### Example

Consider four objects x1, x2, x3, and x4:

x1=unc(10,1,'x1');

x2=unc(20,2,'x2');

x3=unc(30,3,'x3');

x4=unc(40,4,'x4');

Create the array A:

A=[ x1 x2; x3 x4 ];

% A =

%

% 10.0(1.0) 20.0(2.0)

% 30.0(3.0) 40.0(4.0)

The cumulative sum of the elements of the array A is the array B given by:

B = cumsum(A);

% B =

%

% 10.0(1.0) 30.0(2.2)

% 60.0(3.7) 100.0(5.5)

## Jacobian Matrix

### Example

Consider three objects x1, x2, and x3:

x1=unc(7.8333, 3.7103 , 'x1');

x2=unc(3.5000, 1.8708 , 'x2');

x3=unc(12.6667, 4.5898,'x3' );

Consider the following measurement model:

y1= 2 \* x1 - 3 \* x2 ;

y2 = 5 \* x2 + 7 \* x1 ;

y3 = 9 \* x2 - 11 \* x3 + 13 \* x1 ;

Create the vector-valued function Y:

Y=[y1 y2 y3];

The Jacobian matrix of Y can be found using the method [J,Names]= jacobian(Y)

where:

*J*: Jacobian matrix,

*Names*: Names of the variables with respect to which the partial derivatives are taken.

For this particular example:

[J,Names]= jacobian(Y);

% J =

%

% 2 -3 0

% 7 5 0

% 13 9 -11

%

%

% Names =

%

% 'x1' 'x2' []

% 'x1' 'x2' []

% 'x1' 'x2' 'x3'

**Note:** The method Jacobian is not implemented in the UT approach.

## Complex Conjugate Transpose

### Example

Consider four complex objects z1, z2, z3, and z4:

z1 = complex(unc(32.7,1.1),unc(-3.1,0.2)) ;

z2 = complex(unc(11,0.1),unc(-13.1,0.3)) ;

z3 = complex(unc(8,0.2),unc(31.1,0.4)) ;

z4 = complex(unc(7,0.3),unc(3.3,0.1)) ;

Create the matrix Z:

Z=[ z1 z2 ; z3 z4 ];

% Z =

%

% 32.7(1.1) - 3.10(20) \* i 11.00(10) - 13.10(30) \* i

% 8.00(20) + 31.10(40) \* i 7.00(30) + 3.30(10) \* i

The complex conjugate transpose of Z is:

ctranspose(Z)

% ans =

%

% 32.7(1.1) + 3.10(20) \* i 8.00(20) - 31.10(40) \* i

% 11.00(10) + 13.10(30) \* i 7.00(30) - 3.30(10) \*

# Display of Uncertainty Contributions

Consider a an uncertainty object *y* given as a function *f* of some other uncertainty objects x1, x2, …, xn:

y=f(x1, x2, …, xn)

The method *disp\_contribution(y)* displays the contributions of each of the objects x1, x2, …, xn to the uncertainty of y.

## Example

Consider three objects x1, x2, and x3:

x1=unc(7.8333, 3.7103 , 'x1');

x2=unc(3.5000, 1.8708 , 'x2');

x3=unc(12.6667, 4.5898 ,'x3');

Consider the measurement model y:

y = x1 \* x2 - x3;

The contributions of x1, x2, and x3 to the uncertainty of y is displayed using the command:

disp\_contribution(y)

Uncertainty Contribution:

Variable Name | Contribution

-----------------------------

x2 .......... | 14.7

x1 .......... | 13

x3 .......... | 4.59

**Note:** The method disp\_contribution is not available in MC and UT approaches.

# Comparison of Uncertainty Objects

## The greater than Operator

### Comparison of two Uncertainty Objects

The inequality of two uncertainty objects *a* and *b* given by the *greater than* operator > :

a > b

is an uncertainty object unc() where:

: is the probability that a random sample taken from the Gaussian distribution of a is greater than a random sample taken from the Gaussian distribution of b.

### Example:

Consider two objects a and b:

a = unc( 13 , 1 );

b = unc( 11 , 1.5 );

The corresponding Gaussian distributions of a and b are shown in figure 1.

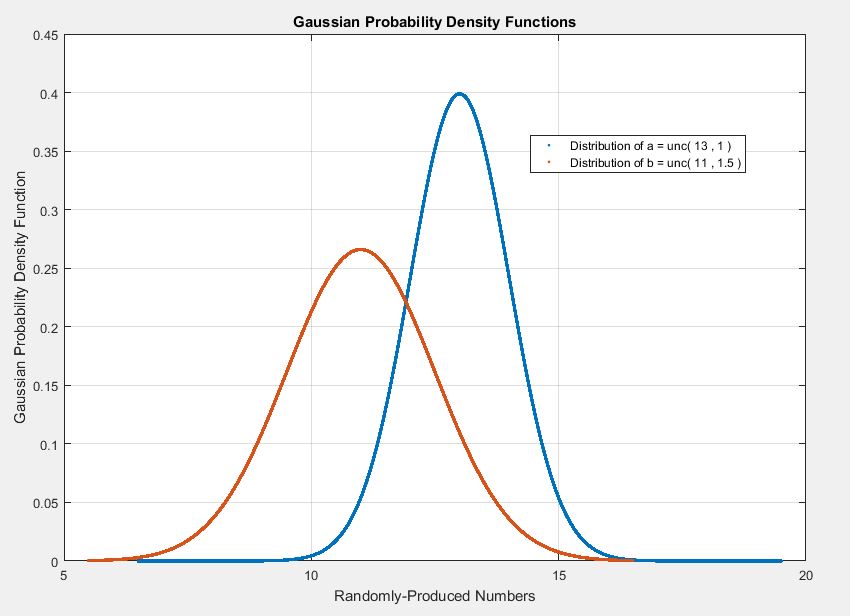


Figure 1: Gaussian Distributions of a and b

The inequality given by:

a > b ;

% ans =

%

% 1.00(87)

means that: the mean value of *a* is greater than the mean value of *b*, and the probability that a random sample taken from the distribution of *a* is greater than a random sample taken from the distribution *b* is 87%.

You can expect that the result of swapping the terms of the inequality would lead to a probability of 100 – 87 = 13%. Indeed,

b > a

% ans =

%

% 0.00(13)

### Comparison of an Uncertainty Object with a Constant Number

The inequality of an uncertainty object *x* and a constant number *c* given by the *greater than* operator > :

x > c

is an uncertainty object unc() where:

: is the probability that a random sample taken from the Gaussian distribution of *x* is greater than the constant number c.

### Example:

Consider an object x and a constant number c:

x = unc( 11 , 1.5 );

c = 10 ;

The inequality given by:

x > c ;

% ans =

%

% 1.00(75)

means that: the mean value of *x* is greater than the constant number c=10, and the probability that a random sample taken from the distribution of x is greater than c=10 is 75%.

You can expect that the result of swapping the terms of the inequality would lead to a probability of 100 – 75 = 25%. Indeed,

c > x ;

% ans =

%

% 0.00(25)

## The less than Operator

### Comparison of two Uncertainty Objects

The inequality of two uncertainty objects *a* and *b* given by the *less than* operator < :

a < b

is an uncertainty object unc() where:

: is the probability that a random sample taken from the Gaussian distribution of a is less than a random sample taken from the Gaussian distribution of b.

### Example:

Consider two objects a and b:

a = unc( 13 , 1 );

b = unc( 11 , 1.5 );

The inequality given by:

b < a ;

% ans =

%

% 1.00(87)

means that: the mean value of *b* is less than the mean value of *a*, and the probability that a random sample taken from the distribution of *b* is less than a random sample taken from the distribution *a* is 87%.

You can expect that the result of swapping the terms of the inequality would lead to a probability of 100 – 87 = 13%. Indeed,

a < b ;

% ans =

%

% 0.00(13)

### Comparison of an Uncertainty Object with a Constant Number

The inequality of an uncertainty object *x* and a constant number *c* given by the *less than* operator < :

x < c

is an uncertainty object unc() where:

: is the probability that a random sample taken from the Gaussian distribution of *x* is less than the constant number c.

### Example:

Consider an object x and a constant number c:

x = unc( 11 , 1.5 );

c = 10 ;

The inequality given by:

x < c ;

% ans =

%

% 0.00(25)

means that: the mean value of *x* is *not less than* the constant number c=10, and the probability that a random sample taken from the distribution of x is less than c=10 is 25%.

You can expect that the result of swapping the terms of the inequality would lead to a probability of 100 – 25 = 75%. Indeed,

c < x ;

% ans =

%

% 1.00(75)

**Note:** The operators >,< are not overloaded in the UT approach.

# Generation of Uncertainty Objects from Data Files

## Working with Excel Files

### Syntax 1: read\_data( file\_name , sheet\_name , range , 'r/c')

Read raw data from a specified range *range* of an Excel sheet *sheet\_name* of an Excel file *file\_name* and generate unc objects either from rows ( if argument r is used) or columns( if argument c is used).

Mean values and standard uncertainties are computed each row (if argument r is used) or column ( if argument c is used) and uncertainty objects are created accordingly.

**Example:**

First, create an uncertainty object that is used to access the static method read\_data():

Tout=unc();

Consider the data file shown in figure 2.

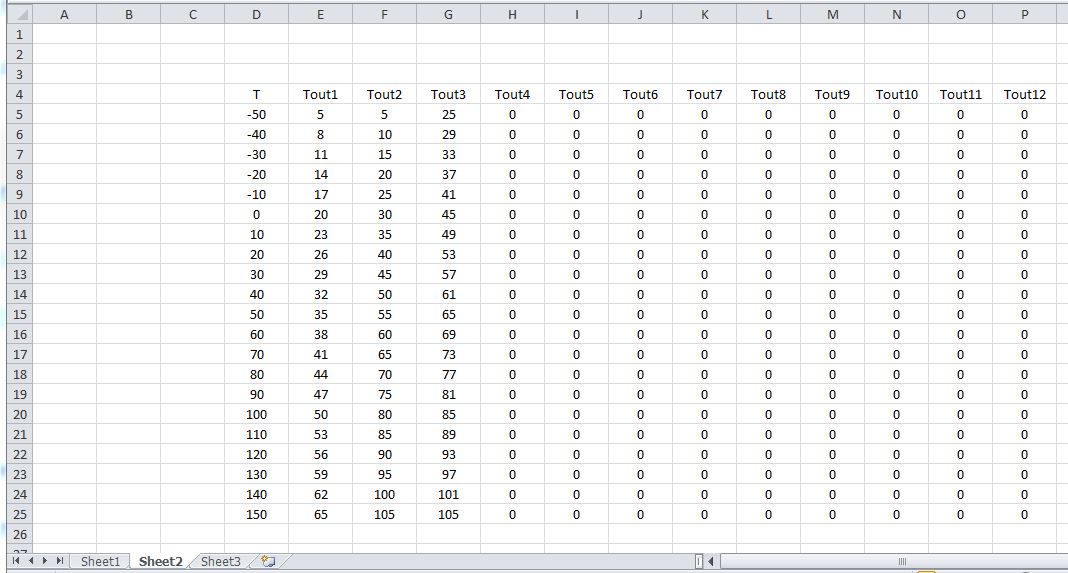


Figure 2: Data File example\_data.xlsx

Generate uncertainty objects from sheet 2 of the Excel file example\_data.xlsx, the desired data being located horizontally in the range E5:P25 :

Tout=Tout.read\_data('example\_data.xlsx', 'Sheet2' , 'E5:P25' , 'r');

% Tout =

%

% 2.9(7.2) 3.9(8.6) 5(10) 6(12) 7(14)

% 8(15) 9(17) 10(19) 11(21) 12(22) 13(24)

% 14(26) 15(28) 16(30) 17(32) 18(33) 19(35)

% 20(37) 21(39) 22(41) 23(43)

### Syntax 2:read\_data(file\_name , sheet\_name , mean\_val , std\_unc , names)

Read data by specifying a range for *mean\_val*,*std\_unc*, and *names*.

**Example:**

First, create an uncertainty object that is used to access the static method read\_data():

Tout=unc();

Consider the data file shown in figure 3.

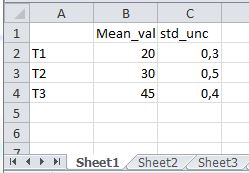


Figure 3: Data File example\_data.xlsx

From the above figure:

The mean values are located in the range 'B2:B4'

The standard deviations are located in the range 'C2:C4'

The names of the variables are located in the range 'A2:A4'

Therefore, the uncertainty objects can be created using the command:

T = T.read\_data('example\_data.xlsx', 'Sheet1' , 'B2:B4' , 'C2:C4' , 'A2:A4');

% ans =

%

% 20.00(30)

% 30.00(50)

% 45.00(40)

Check for the name of a variable, for example:

T(2).name

% ans =

%

% 'T2'

### Syntax 3: read\_data( file\_name , sheet\_name , mean\_val , std\_unc )

Read data by specifying range for *mean\_val* and *std\_unc.*

In this case, no names are assigned in the data file to the variables.

**Example:**

First, create an uncertainty object that is used to access the static method read\_data():

data=unc();

Generate uncertainty objects from the Excel file given in the Figure 4.

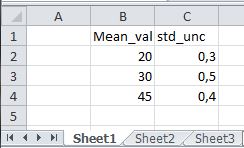


Figure 4: Data File example\_data.xlsx

data.read\_data('example\_data.xlsx', 'Sheet1' , 'B2:B4' , 'C2:C4')

% ans =

%

% 20.00(30)

% 30.00(50)

% 45.00(40)

## Working with CSV and DAT Files

### Syntax 1: read\_data( file\_name , 'r/c')

Read data from the file *file\_name*. Mean values and standard uncertainties are computed each row (if argument r is used) or column ( if argument c is used) and uncertainty objects are created accordingly.

**Example 1:**

First, create an uncertainty object that is used to access the static method read\_data():

Tout=unc();

Generate uncertainty objects from the rows of the dat file given in figure 5:

Tout=Tout.read\_data('example\_data.dat' , 'r');

% Tout =

%

% 2.9(7.2) 3.9(8.6) 5(10) 6(12) 7(14) 8(15) 9(17)

**Example 2:**

First, create an uncertainty object that is used to access the static method read\_data():

Tout=unc();

Generate uncertainty objects from the columns of the dat file given in figure 5:

Tout=Tout.read\_data('example\_data.dat' , 'c');

% Tout =

%

% 14.0(6.5) 20(11) 37.0(8.6) 0(0) 0(0) 0(0) 0(0) 0(0) 0(0) 0(0) 0(0) 0(0)

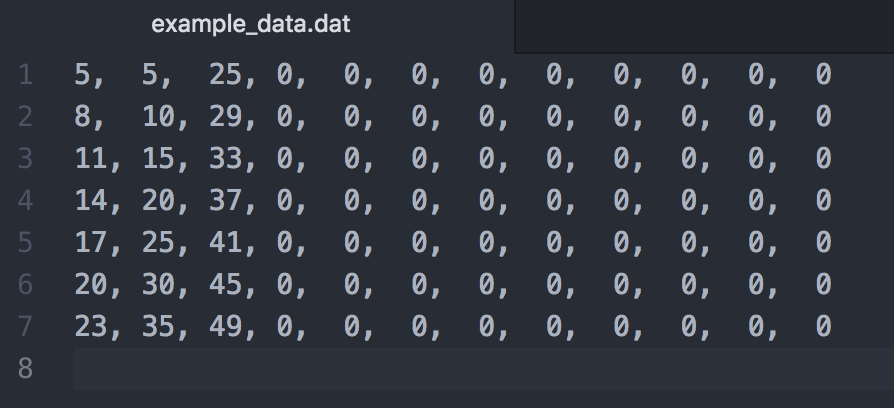


Figure 5: Data File example\_data.dat

**Example 3:**

First, create an uncertainty object that is used to access the static method read\_data():

Tout=unc();

Generate uncertainty objects from the rows of the csv file given in figure 6:

Tout=Tout.read\_data('example\_data.csv' , 'r');

% Tout =

%

% 2.9(7.2) 3.9(8.6) 5(10) 6(12) 7(14) 8(15) 9(17)

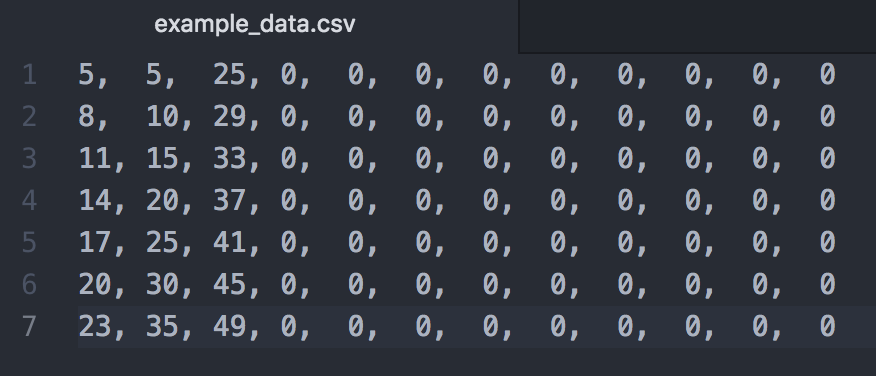


Figure 6: Data File example\_data.csv

### Syntax 2: read\_data( file\_name , R , C , [R C R2 C2] , 'r/c' )

From the file *file\_name*, read only the range specified by [R C R2 C2] where (R,C) is the upper-left corner of the data to be read and (R2,C2) is the lower-right corner.

Mean values and standard uncertainties are computed each row (if argument r is used) or column ( if argument c is used) and uncertainty objects are created accordingly.

**Example:**

First, create an uncertainty object that is used to access the static method read\_data():

Tout=unc();

Generate uncertainty objects from the rows of the dat file given in figure 5:

Tout=Tout.read\_data('example\_data.dat' , 2,2,[2 2 4 4],'r');

% Tout =

%

% 11(19) 12(21) 14(24)

**Note:** The functions implemented in this section are not implemented in the UT approach.

# References

[1] BIPM. JCGM 100:2008: Guide to the expression of uncertainty

in measurement, 2008.

[2] BIPM. JCGM 101:2008: Evaluation of measurement data - supplement 1 to the "guide to the expression of uncertainty in measurement" - propagation of distributions using a monte carlo method evaluaton of measurement data.

[3] H. Zangl and G. Steiner. Optimal design of multiparameter multisensor systems. IEEE Transactions on Instrumentation and Measurement, 57(7):1484–1491, July 2008.

[4] Hubert Zangl, Klaus Hörmaier, “Educational Aspects of Uncertainty Calculation with Software Tools”

[5] Wolfgang Granig, Slimane Aoudjit, Lisa Marie Faller, Hubert Zangl, “Statistical Modeling of Integrated Sensors for Automotive Applications”

[6] MATLAB R2016b, The MathWorks, Inc., Natick, Massachusetts, United States

# Index of Commands

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | abs 52  acos   * Real 37 * Complex 57   acosh   * Real 40 * Complex 61   acot   * Real 38 * Complex 58   asin   * Real 37 * Complex 57   asinh   * Real 39 * Complex 61   atan   * Real 38 * Complex 58   atan2 38  atanh   * Real 40 * Complex 61   Complex 47,48,62  disp\_contribution 76  disp\_dep 9  exp   * Real 35 * Complex 54   get\_cov\_mat 19  get\_cor\_mat 22  get\_input\_cov\_mat 24  get\_input\_cor\_mat 24  gmn 9  gmu 9  gmv 9  grad 10  gt ( > ) 77,78  imag 11  img 11  inv 70  jacobian 75  karhloev   * Real 43,44,45 * Complex 67,69   sinh   * Real 38 * Complex 58   std\_unc 9  sqrt   * Real 40 * Complex 62   sum 73  tan   * Real 37 * Complex 56   tanh   * Real 39 * Complex 60   trace 72  unc 12,13,14,15,16  value 9 | conj 52  cor\_into\_cov\_mat 26  cov\_into\_cor\_mat 25  cos   * Real 36 * Complex 55   cosh   * Real 39 * Complex 59   cot   * Real 37 * Complex 57   coth   * Real 39 * Complex 60   covr 17,41  correl 18  ctranspose 75  cumsum 74  dep 9  det 70  diag 71  log   * Real 35 * Complex 53   log10   * Real 35 * Complex 54   lt ( < ) 79,80  max 72  min 73  name 9,28  OutDataTypeStr 10  r 11  read\_data 81,82,83,84,85  real 11  rel\_mat 10  rel\_obj 10  set\_correl 18,28  set\_cov 17,30  sin   * Real 36 * Complex 56 | |  |
|  |  |
|  |  |
|  |  |