"An Uncertainty Toolbox". Quick Start Guide.

Alpen-Adria-Universitaet (AAU) Klagenfurt
Insitute of Smart Systems Technologies
Sensors and Actuators (SAT).
c/o Hubert Zangl (`hubert.zangl@aau.at`*)*
Universitaetstrasse 65-67, A-9020 Klagenfurt, Austria.

November 8, 2019

Introduction

"An Uncertainty Toolbox"(AUT) is a software tool for MATLAB, developed to perform uncertainty calculations in Measurement Science. The toolbox includes three different approaches: Linearization, Monte Carlo(MC) and Unscented Transformation(UT). The former is based on the GUM (Guide to the expression of uncertainty in measurement [1]) method, where 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, MC method addressed in [2], provides an adequate alternative to estimate the output quantity and its associated standard uncertainty. The latter represents a new uncertainty analysis approach based on the principle of the UT,explained in [3], which is also suitable when linearization of the model provides an inadecuated representation and could involve less calculation than MC, in many cases.

The toolbox is also intended to be useful for system analysis and optimization, provinding a simple way to switch between the aforementioned methods. This guide introduce some simple and practical examples, providing users a quick familiarization with the toolbox.

Contents

1 Defining and reporting uncertainty objects.

To create an uncertain variable we should define its estimated valued and standard uncertainty, as shown Figure 1. Uncertain variables can be interpreted as realization of random variables, where the estimated valued and uncertainty represent the mean and standard deviation, respectively. For reporting the standard uncertainty two significant digits are used in terms of the least significant digits of the estimated value, following the recommendation given in [1].

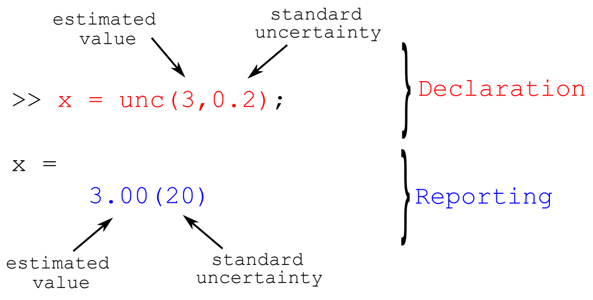


Figure 1: Defining and reporting an uncertain variable x.

Key Note1 : All 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.

Key Note2 : The number of MC samples is set by default to 10000. To change this values reset the global variable MCSAMPLES.

1.1 Demo

This introductory example shows the difference between defining and assigning uncertain variables. Creating variables using *unc* generates independent variables, whereas an assignment set fully correlation between them. Figure 2 depicts how the difference between two variables with equal value and uncertainty is zero with non-zero uncertainty, in case they are defined independently. However, the uncertainty of their difference vanishes when they are assigned (a=b), i.e they are identical. The same applies to ratio. (See MATLAB script **demo.m** for step by step explanation.)

1.2 Cartesian to Polar Coordinate Transformation

In order to understand the differences between the three approaches for uncertainty evaluation, the simple example of converting a point from cartesian coordinates (x, y) to polar coordinates(r, θ) is presented.

```
>>a=unc(100,1);    >>a=unc(100,1);    >>a=unc(100,1);    >>a=unc(100,1);
>>b=unc(100,1);    >>b=a;             >>b=unc(100,1);    >>b=a;
>>c=a-b            >>c=a-b            >>c=a/b            >>c=a/b
c=                 c=                 c=                 c=

   0.0(1.4)           0(0)              1.000(14)          1(0)
```

Figure 2: Introductory example. Creating and assigning uncertain variables.

The Cartesian coordinates are defined as uncertain variables with mean 0.01 and standard uncertainty 0.005.(See MATLAB script **CartesiantoPolar.m**) The radial and angular coordinates are computed by $r = \sqrt{x^2 + y^2}$ and $\theta = \arctan(y/x)$ respectively. Figure 3 shows a comparison of the first two moments (mean and variance) when Linearization, MC and UT are performed. Notice that Linearization leads to errors in estimating the mean and variance in polar coordinates, while UT (5 sigma points) gives a result closer to MC(10^6).

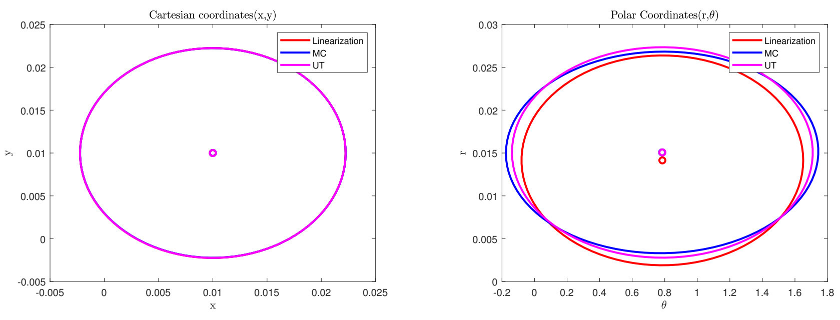


Figure 3: Comparison of mean and covariance ellipse (3σ) in Cartesian and Polar planes.

2 Application examples

2.1 Hall sensor

Considering the equation for a Hall voltage $U_{gem} = VSIB$ (more details in [4]), we can determine the flux density B from the Hall voltage under consideration of the random sensor offset U_{oH} and random amplifier offset U_{oIV} by

$$B = \frac{\frac{U_{gem}}{V} - U_{oIV} - U_{oH}}{IS} \tag{1}$$

Figure 4 depicts how to define the input parameter using the toolbox. Note that names are assigned to each parameter, so they can be easily identified. The contribution of the uncertain input variables to the combined uncertainty of the result can be accessed by the method *disp_contribution*.

```
            value    std_unc                 name
I=unc(1e-3,1e-5,'Bias Current');
U_oIV=unc(0,1e-3,'Amplifier Offset');
U_oH=unc(0,1e-3,'Hall Sensor Offset');
V=unc(100,1,'IV Amplification');
S=unc(2e-3,1e-4,'Hall Sensor Sensitvity');
B_K=unc(0,3,'Calibration Flux Density');
```

Figure 4: Defining inputs parameters of Hall Sensor.

After computation of the flux density using equation (1) it can be seen that the major contribution results from the offsets(See Figure 5). However, as all output voltages are subject to the same offsets, an offset calibration can be used. Figure displays the result for the flux density with and without calibration. MATLAB script **Hall.m** describes the example step by step.

```
>> B=(Ugem/V-U_oIV-U_oH)/I/S;
>> disp_contribution(B(10))

Uncertainty Contribution:
Variable Name           | Contribution
---------------------------------------
Amplifier Offset ...... | 500
Hall Sensor Offset .... | 500
Hall Sensor Sensitvity  | 7.47
IV Amplification ...... | 1.49
Bias Current .......... | 1.49
ADC ................... | 0.5
```

```
>> BK=B(:)-B(1)+B_K;
>> disp_contribution(BK(10))

Uncertainty Contribution:
Variable Name              | Contribution
----------------------------------------
Calibration Flux Density   | 3
Hall Sensor Sensitvity ..  | 2.47
ADC ...................... | 0.5
ADC ...................... | 0.5
IV Amplification ........  | 0.494
Bias Current ............  | 0.494
```

Figure 5: Determination of flux density and its contribution with and without calibration

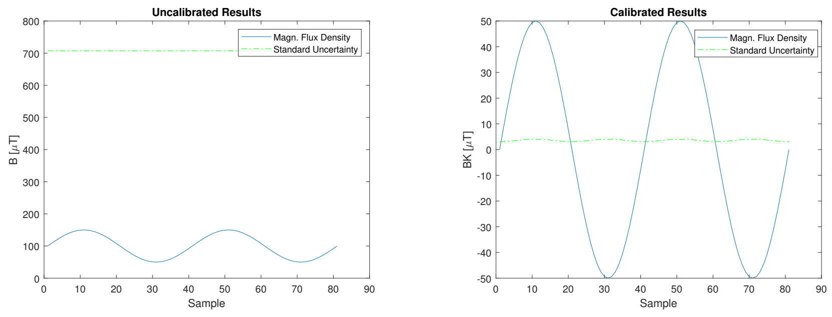


Figure 6: Measurement results for the flux density with and without calibration.

2.2 Maxwell-Wien Bridge

The Maxwell-Wien bridge is an extension of the Wheatstone Bridge circuit towards the complex domain. It is found to be more suitable for measuring unknown inductance (usually with low quality factor) and its circuit is shown in Figure 7. [5]

Figure 8 exemplifies how to define an uncertain complex quantity Z.

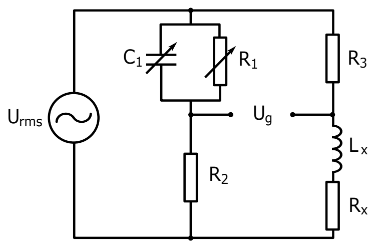


Figure 7: Example of Maxwell-Wien bridge.

First, the real and imaginary components are declared independently and then the function *complex* is used.

```
>> x = unc(3,0.2,'real_Z');
>> y = unc(4,0.1,'imag_Z');
>> Z = complex(x,y)
Z =
      3.00(20) + 4.00(10) * i
```

Figure 8: Defining a complex uncertain variable Z.

Instead of using the classical balance equation to determine the unknown impedance $Z_x = R_x + jwL_x$, a more complete measurement model is employed, which takes into account the influences introduced by the choice of U_{rms} and the accuracy of the instrument that measures U_g. Then Z_x is given by

$$Z_x = \frac{U_{rms} Z_3 (Z_1 + Z_2)}{U_{rms} Z_1 + U_g (Z_1 + Z_2)} - Z_3 \tag{2}$$

Figures below show the procedure to calculate Z_x (See **AC_Bridge.m**) and how to obtain the covariance and correlation matrices associated with R_x and L_x using the functions *get_cov_mat* and *get_cor_mat*.

```
>> Urms = unc(10,5,'Urms'); Ug = unc(0,0.01,'Ug');
>> R1 = unc(1.001e3,1,'R1'); C1 = unc(15.e-6,1e-6,'C1');
>> R2 = unc(1.001e3,3,'R2'); R3 = unc(1.001e3,3,'R3');
>> f = 50; w = 2*pi*f;
% Z1 = R1||C1 = R1 /( 1+j*w*R1*C1 )
>> realZ1 = R1 /(1+(w*R1*C1)^2);
>> imagZ1 = - w*C1*R1^2/(1+(w*R1*C1)^2);
>> Z1 = complex(realZ1, imagZ1);
>> Z2 = R2;
>> Z3 = R3;
>> Zx = Urms*Z3*(Z1+Z2)/(Urms*Z1 + Ug*(Z1+Z2)) - Z3
Zx =
     1001.0(19) + 4720(320) * i
>> Rx = real(Zx)
Rx =
     1001.0(19)
>> X_L = imag(Zx);Lx = X_L/w
Lx =
     15.0(1.0)
```

Figure 9: Matlab code to calculate the unknown impedance Z_x and the corresponding resistance R_x and inductance L_x.

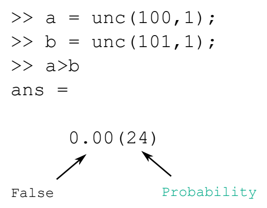
```
>> Cov_Zx = get_cov_mat([Rx X_L])  >> Cor_Zx = get_cor_mat([Rx X_L])
Cov_Zx =                           Cor_Zx =

   1.0e+04 *                          1.0000   -0.0438
                                     -0.0438    1.0000
    0.0353   -0.0260
   -0.0260    9.9849
```

Figure 10: Determination of the Covariance and Correlation matrices of the vector $\mathbf{Z}_x$ using the uncertainty toolbox.

2.3 Probability of Bit Error with Increasing Distance

Comparisons (such as needed in detectors) based on uncertain variables return uncertain results. The toolbox returns the probability of returning the correct result assuming Gaussian distributions with a standard deviation corresponding to the standard uncertainty.

The basic use[1] is illustrated in Figure 11.



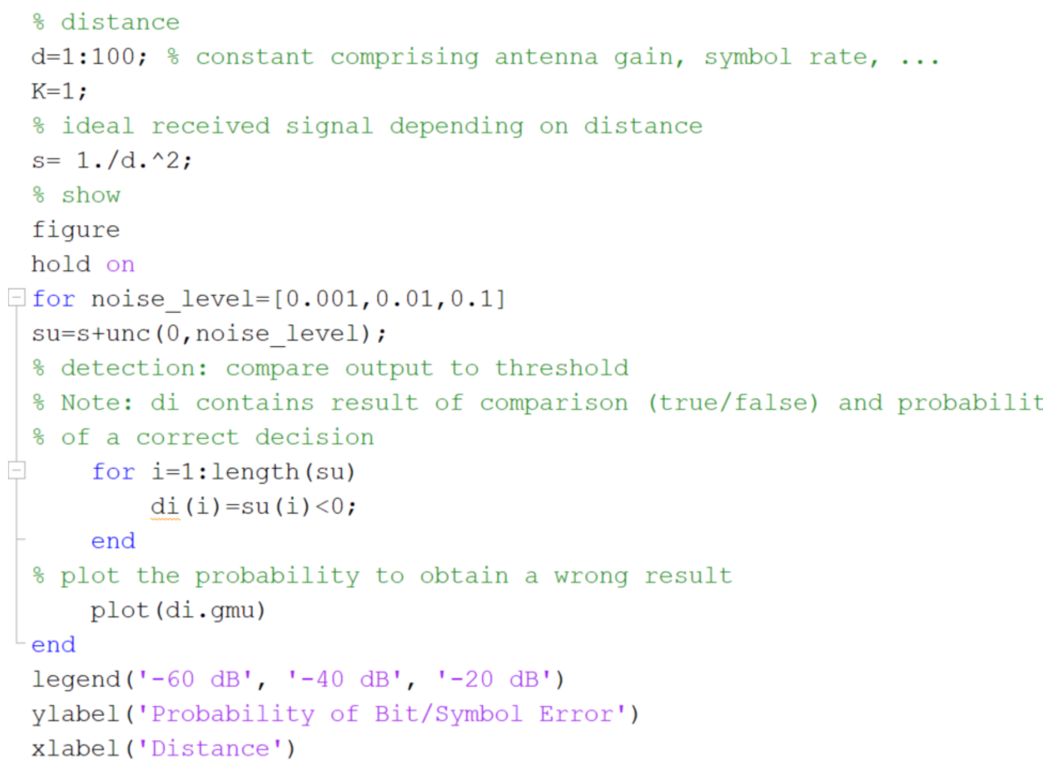
```
>> a = unc(100,1);
>> b = unc(101,1);
>> a>b
ans =

     0.00(24)
```

Figure 11: Comparison with uncertain variables. 0.00 corresponds to "False" and 1.00 corresponds to "True". The value in brackets is the corresponding probability, i.e. the probability to decide for a<b even though a>b.

An example for analyzing the performance of a receiver with increasing distance form the transmitter is given in Figure 12.

The corresponding result is shown in Figure 13.

[1] In the current version of the toolbox, the result is returned as an uncertain object. Please note that thus the current version does not support logical operations. In future versions, it is planed to additionally define uncertain logical variables.



```
% distance
d=1:100; % constant comprising antenna gain, symbol rate, ...
K=1;
% ideal received signal depending on distance
s= 1./d.^2;
% show
figure
hold on
for noise_level=[0.001,0.01,0.1]
su=s+unc(0,noise_level);
% detection: compare output to threshold
% Note: di contains result of comparison (true/false) and probabilit
% of a correct decision
    for i=1:length(su)
        di(i)=su(i)<0;
    end
% plot the probability to obtain a wrong result
    plot(di.gmu)
end
legend('-60 dB', '-40 dB', '-20 dB')
ylabel('Probability of Bit/Symbol Error')
xlabel('Distance')
```

Figure 12: Simple example for analyzing the performance of a receiver over the distance from a transmitter using different signal power.

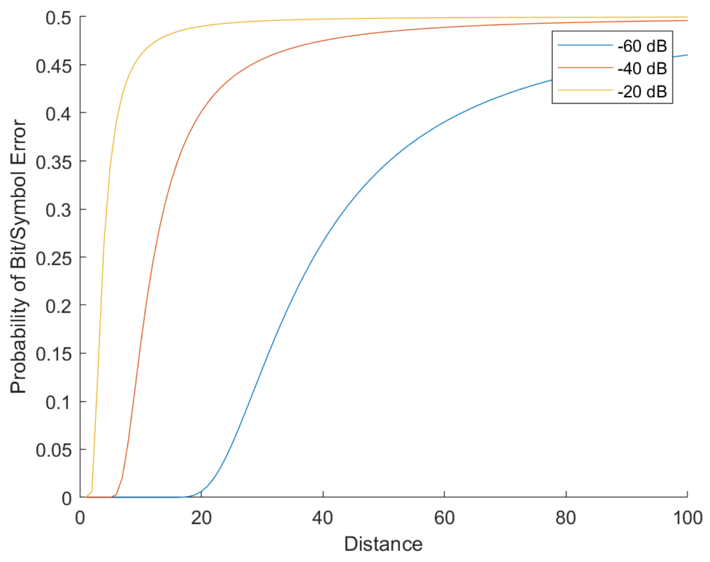


Figure 13: Result of the code according to Figure 12. With decreasing output power of the transmitter, the performance over distance gets significantly worse. A probability of 0.5 corresponds to purely random detection results.

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 and Klaus Hoermaier. Educational aspects of uncertainty calculation with software tools. *Measurement 101*, pages 257–264, 2017.

[5] Dailys Arronde Pérez and Hubert Zangl. Introducing uncertainty of complex-valued quantities in measurement science education. In *Photonics and Education in Measurement Science 2019*, volume 11144, pages 247 – 253. International Society for Optics and Photonics, SPIE, 2019.