User manual: Sensitivity analysis and Bayesian calibration for ECN Aero-Module

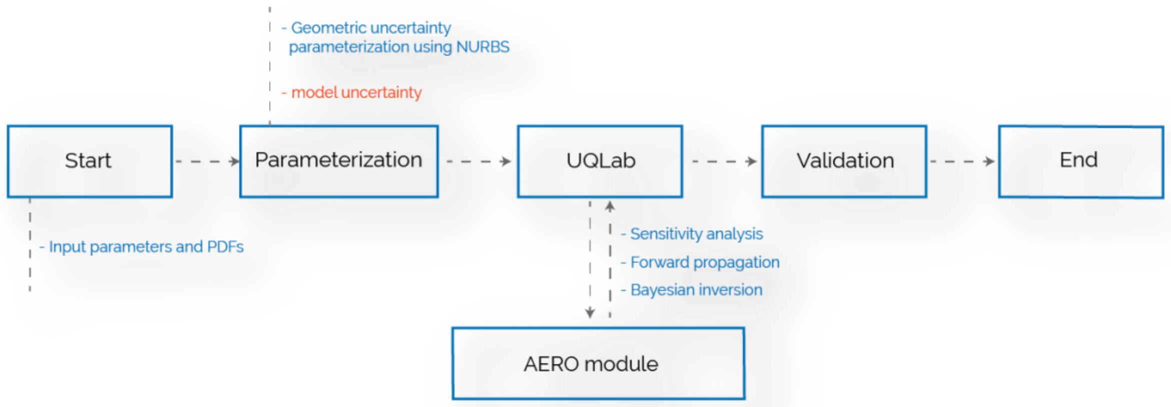
0. Background/introduction

In the WindTrue project, uncertainties in aeroelastic wind turbine models are studied. The first step in the project is to study which parameters are the most important given certain quantities of interest (e.g. power, forces, moments).

The sensitivity analysis as performed here is built on two main ingredients:

1. The computational model representing the wind turbine: **ECNAero**. ECNAero is a software code developed and maintained by ECN part of TNO.
2. The uncertainty quantification framework: **UQLab**. UQLab is a Matlab-based uncertainty quantification package developed and maintained at ETH Zürich and free to use for academic purposes.

The sensitivity analysis code presented here combines these two tools by using smart parameterizations (input processing) and wrappers to process the output of ECNAero.



1. Installation

The module can be installed by downloading the directory **sensitivity_analysis** from the windtrue git repository [https://github.com/bsanderse/windtrue.git].

The **sensitivity_analysis** directory contains the necessary routines for sensitivity analysis: uncertainty parameterization routines developed at CWI, Aero module routines [in the directory **AEROmodule**] and UQLab routines [in the directory **UQLab**].

System specifications
Operating system: Windows 10
Matlab: R2019a and further
UQLab: 1.3.0
ECNAero: v252

The UQLab installation present in the UQLab directory can be directly used. However, it can happen that a newer UQLab version is available, which can be installed instead of the current available version on our Windtrue GitHub. A newer version can be downloaded from the UQLab website and is typically named as UQLabCore_Rel1.x.x.Zip. This file should be unzipped inside the directory **sensitivity_analysis**/**UQLab/**. The UQLab software requires a license, which can be obtained from the UQLab website [https://www.uqlab.com/]. The license file (that a user will obtain via email from the UQLab administrators) should be placed inside the directory **sensitivity_analysis**/**UQLab/core**. To activate the license and install UQLab package from the Matlab command window, go to the directory **sensitivity_analysis**/**UQLab/core** and run the command *uqlab_install*.

2. Executing the Sensitivity Analysis for a wind turbine

2.0 Overview

The core of the sensitivity analysis is the file **testSensitivity.m**. Once this file is executed by the user, UQLab is started, several ECNAero evaluations are performed, and output is post-processed and visualized.

Several files can be changed in order to change the Sensitivity Analysis:

- **testSensitivity.m**: specify the case that will be executed, the type of Sobol analysis, and the postprocessing (see 2.4).
- **cases/aero_module/initialize.m**: details of the methods used to perform the analysis, e.g. number of samples, polynomial degree, etc (see 2.3).
- **cases/aero_module/NM80.m**: specification of the uncertain input variables (see 2.1).

2.1 Setting up the turbine data routine

Turbine and site related data as well as the probability distributions for different random variables are defined in the routine **NM80.m** that is stored in the directory **sensitivity_analysis/cases/aero_module/.** The variable names in **NM80.m** are adopted from the **input.txt** file of the AEROmodule software.

The properties of the uncertain parameters are defined as a struct variable in Matlab which is similar to the random variable definition in UQLab (e.g. `Input.Marginals.XYZ`). This routine already contains definitions for most of the uncertain parameters. New parameters can be simply added as follows:

```
counter = counter+1;
Input.Marginals(counter).Name = 'NewParameter';
Input.Marginals(counter).Index = ''; % For indexing vectors. Set empty for scalars
Input.Marginals(counter).Type = 'Uniform'; % Any distribution available in UQLab
Input.Marginals(counter).Parameters = [Mean, Std];
Input.Marginals(counter).Bounds = [LowerBound, UpperBound];
```

For a vector parameter set, we have to define uncertainty for each element of the vector, as is done for example in the definition of uncertainty in the twist along a blade. A similar approach can be used as for the chord/twist/thickness curve defined already in this routine.

Although we define properties for all uncertain parameters, we can selectively perform sensitivity analysis using only a subset of these parameters. The names of these selected parameters are specified in the variable `uncertain_params` at the end of the file.

The routine **getParameterAeroModule.m** also located in the directory **sensitivity_analysis/cases/aero_module/** calls the **NM80.m** routine and converts the turbine/site data and uncertainty definitions in a format that is suitable for the UQLab routines. All the data from the **NM80.m** routine is put into in the cell variable `P{}`. Further, if a new parameter is added to the **NM80.m** file then the **getParameterAeroModule.m** routine should also be updated by appending the new parameter at the end of the cell variable `P{}`. Note:

The order in which the input parameters are assigned to `P{}` should not be changed, and new elements shall only be added in the end.

2.2 Writing the sampled random parameters in the AEROmodule

UQLab generates samples of random variables that should be written to AEROmodule input file **input.txt**. For this, we use the **writeAeroModuleInput.m** routine located in the directory **/sensitivity_analysis/cases/aero_module/.** This routine modifies the **input.txt** file using the sampled random values of the parameters. If a new parameter is added, this routine should be modified accordingly.

2.3 Setting up the properties of Sensitivity analysis algorithm

One can specify the algorithms used to compute the Sobol indices in **initialize.m** routine located in the directory **sensitivity_analysis/cases/aero_module/**.

There are currently four possible algorithms to compute Sobol indices: Monte Carlo (MC), Polynomial Chaos Expansion based on quadrature (PCE_Quad), Polynomial Chaos Expansion based on Ordinary Least Squares and Polynomial Chaos Expansion based on Least Angle Regression (PCE_LARS). For details, please see the UQLab manual.

We specify a list of the methods that we request UQLab to run for example as follows:

```
methods = {'MC','PCE_Quad','PCE_OLS','PCE_LARS'};
```

For each method, we further describe the number of samples or polynomial order (this depends on the method type) as:

```
NsamplesMC = [10 100 1000]; % Run 3 different simulations with 10,100,1000 samples
MC_repeat = 1; % To repeat the experiments a number of times

% For PCE-Quad, specify the polynomial degrees to be tested
DegreesQuad = 1:4; % this will have polynomial up to degree 4

% For PCE-OLS, if not specified, number of samples from PCE-Quad is used
NsamplesOLS = [16 32 64]; % Run 3 different simulations with 16,32,64 samples
OLS_repeat = 1; % To repeat the experiment a number of times

% For PCE-LARS, if not specified, number of samples from PCE-Quad is used
NsamplesLARS = [16 32 64]; % Run 3 different simulations with 16,32,64 samples
LARS_repeat = 1; % To repeat the experiment a number of times
```

All four methods can be used to perform comparison and to verify whether the approaches result in similar sensitivities indices. In practice, often only a single algorithm is used (because of the computational expenses of running all algorithms can be high). For example, when we want to use the PCE_LARS algorithm, we use the following:

```
methods = {'PCE_LARS'};
NsamplesLARS = [128];
LARS_repeat = 1; % Without repetition
```

Other variables can be commented out.

2.4 Running the sensitivity analysis

The sensitivity analysis can be started by running **testSensitivity.m** routine located in the **sensitivity_analysis** directory. The output is a histogram plotting the values of Sobol indices for each random parameter.

2.5 Output description

The histogram contains the values from variable `AVG_Sobol_XYZ_Total`, for example, `AVG_Sobol_MC_Total`, which stores the average values of total Sobol indices of each random variable, computed using MC method. The average is based on `MC_repeat` number of experiments. Similarly, `AVG_Sobol_OLS_Total` stores the average value of total Sobol indices computed using the OLS (ordinary least square) method.

Also, the mean and standard deviations of the quantity of interest is stored in the variables `mean_LARS` and `std_LARS`, respectively if PCE_LARS is used for is used for sensitivity analysis.

Discuss about two examples NM80 and AVATAR
Explain how to parameterize the chord, twist, etc. [A priori analysis]

3. Executing Bayesian analysis for ECN Aero-Module

The core of Bayesian calibration lies in the file **testCalibration.m**. Once this file is executed by the user, UQLab is started, calibration is performed, and output is post-processed and visualized.

The following files need to be addressed for setting up the Bayesian calibration:

3.1 NM80_calibrate.m

Turbine and site related data as well as the probability distributions for different random variables are defined in the routine **NM80_calibrate.m**, which is stored in the directory **sensitivity_analysis/cases/aero_module/.**

The **NM80_calibrate.m** routine is identical to **NM80.m** used for sensitivity analysis study (refer to section 2.1 for detailed description)

Although we define properties for all uncertain parameters, we can selectively perform calibration for only a subset of these parameters. The selection is specified in the variable **uncertain_params** at the beginning of the file. For example, when we want to calibrate the lift coefficient polars, we use the following:

```
uncertain_params = {{'CL',1, 0.2}, {'CL',2, 0.2}, {'CL',3, 0.3},{'CL',4,0.3}};
```

Since, calibration would require comparison of the Aero-Module output with the experimental measurements, an additional Aero-Module output file is stored in the new cell variable `P{44}`(see **getParameterAeroModule_cal.m**). Below, the normal force data obtained from the Aero-Module run will be used to performing calibration.

```
QoI = 'force'; % Force or Power at different radial stations
aero_module_outputfile = 'B1n_BEM.txt'; % Force at different radial stations
```

Further, if a new parameter is added to the **NM80_calibrate.m** file then the **getParameterAeroModule.m** routine should also be updated by appending the new parameter at the end of the cell variable `P{}`.

3.2 initialize_calibration.m

The key ingredients to perform the Bayesian calibration are defined in this routine, which is stored in the directory **cases/aero_module/initialize_calibration.m**

- Forward model: The forward model, i.e. Aero-Module, is defined as a MATLAB m-file which is stored in the folder: **cases/aero_module/aero_module_calibration.m**. This routine runs the **ECNAero.exe** executable, and later stores the output data in handle **'Y'**.
- Experimental data: The experimental measurements are stored in handle **'y'.** Currently, two options can be passed on for reading the DANAERO data:
    - raw_normal.dat for reading the time series normal force measurements
    - raw_tangential.dat for reading the time series tangential force measurements

  Because the forward model has different discrepancy options at different radial locations, the measurement data is stored in four different data structures **[Data(1).y, Data(2).y, Data(3).y, Data (4).y]**
- Discrepancy/Likelihood: The discrepancy defines the connection between the measurement data and the forward model data. We assume that the discrepancy (σ^2) is not known a priori. Then, the Bayesian framework can infer the σ^2 distribution of the discrepancy parameter with the initial specification of the prior distribution. For example, the prior distribution for data structure **Data(1).y** is specified as:

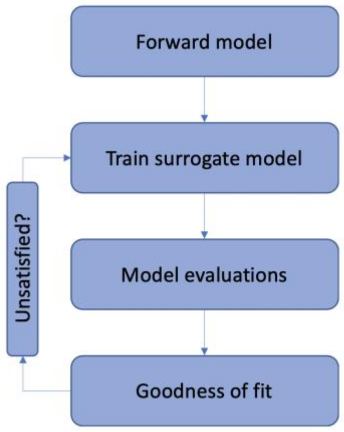
```
DiscrepancyPriorOpts1.Name = 'Prior of sigma 1';
DiscrepancyPriorOpts1.Marginals(1).Name = 'Sigma1';
DiscrepancyPriorOpts1.Marginals(1).Type = 'Uniform';
DiscrepancyPriorOpts1.Marginals(1).Parameters = [0, 2*std(output_raw.Fy03)];
DiscrepancyPrior1 = uq_createInput(DiscrepancyPriorOpts1);

DiscrepancyOpts(1).Type = 'Gaussian';
DiscrepancyOpts(1).Prior = DiscrepancyPrior1;
```

  In the present case, four independent prior distributions $\pi(\sigma^2)$ have been specified. The groups are defined by specifying the **DiscrepancyOpts** as structure arrays.

- Prior distribution: The prior distributions of the Aero-Module parameters are defined as an INPUT object in **NM80_calibrate.m**. For example, the prior distribution of the lift coefficient polar corresponding to **Data(1).y** reads:

```
% Define PDF for CL(1)
counter = counter+1;
Input.Marginals(counter).Name = 'CL';
Input.Marginals(counter).Index = 1;
Input.Marginals(counter).Type = 'Uniform';
Input.Marginals(counter).Parameters = [-1.5 1.5];
Input.Marginals(counter).Bounds = [-1.5 1.5];
```

- Forward model options: Training a surrogate model comprises of the following steps:



The switch between the full forward model and the surrogate model is specified as follows:

```
Bayes_full = 0; % 0: use surrogate model (PCE); 1: run full model for Bayes
(Computationally expensive!)
Surrogate_model_type = 0; % 0: Uses a stored PCE surrogate model, 1: create
surrogate model
Surrogate_model_filename = 'surrogate/PCE_LARS.mat'; % Specify the surrogate
model file to be used
```

Currently, 2 surrogate models PCE_LARS_normal.mat and PCE_LARS_tangential.mat are trained for performing calibration using the normal force and tangential force data respectively. They are stored in the **surrogate** folder. It is noteworthy that the saved surrogate models can be treated as digital twins in lieu of the original forward model.

The code needed to create a basic LARS-based PCE surrogate model with 100 sample points of full model evaluations is as follows:

```
MetaOpts.Type = 'Metamodel';
MetaOpts.MetaType = 'PCE';
MetaOpts.Method = 'LARS'; % Quadrature, OLS, LARS

MetaOpts.ExpDesign.Sampling = 'LHS';
MetaOpts.ExpDesign.NSamples = 100;
MetaOpts.Degree = 1:4;
MetaOpts.TruncOptions.qNorm = 0.75;
```

Note: For the current case, 100 sample points were sufficient to create a surrogate model with an estimated LOO error of 10^{-6}. This may change on case-to case basis, leaving the   user to try different number of sample points to create a surrogate model.

A trained surrogate model can be saved using the following syntax in the command window: save surrogate/**file_name**.mat mySurrogateModel

- MCMC options: Currently, four MCMC algorithms are present to perform MCMC sampling using UQLAB: Metropolis Hastings (MH), Adaptive-Metropolis (AM), Hamiltonian Monte Carlo (HMC) and Affine Invariant Ensemble Sampler (AIES). The number of iterations is specified as a scalar in the **.Steps** field and the number of chains by passing a scalar value to the **NChains**

```
Solver.MCMC.Steps = 1e2;
Solver.MCMC.NChains = 1e2;
```

3.3 testCalibration.m

The Bayesian calibration can be started now by running **testCalibration.m** routine located in the **sensitivity_analysis** directory.

3.4 PostProcessingCaibration.m

All post-processing results are generated by the **uq_postProcessInversion** function are stored in the **BayesianAnalysis.Results.PostProc** structure
Location: **cases/aero_module/PostProcessingCaibration.m**