UQLab: the uncertainty quantification software framework

Bruno Sudret

Chair of Risk, Safety and Uncertainty Quantification

ETH Zurich

Weimar, September 5th, 2016

Computational models in engineering

Complex engineering systems are designed and assessed using computational models, a.k.a simulators

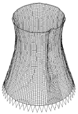
A computational model combines:

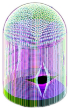
- A mathematical description of the physical phenomena (governing equations), *e.g.* mechanics, electromagnetism, fluid dynamics, etc.

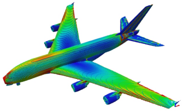
$$\begin{aligned} \nabla \cdot \mathbf{D} &= \rho \\ \nabla \cdot \mathbf{B} &= 0 \\ \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{H} &= \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \end{aligned}$$

- Discretization techniques which transform continuous equations into linear algebra problems

- Algorithms to solve the discretized equations

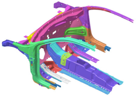




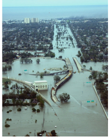


Real world is uncertain

- Differences between the designed and the real system:
  - Dimensions (tolerances in manufacturing)
  - Material properties (*e.g.* variability of the stiffness or resistance)



- Unforecast exposures: exceptional service loads, natural hazards (earthquakes, floods, landslides), climate loads (hurricanes, snow storms, etc.), accidental human actions (explosions, fire, etc.)



What is uncertainty quantification ?

Input Variables	→	Computational Model	→	Quantities of Interest
$\boldsymbol{X} \in \mathcal{D}_{\boldsymbol{X}}$		$\mathcal{M}$		$Y = \mathcal{M}(\boldsymbol{X})$

- What is the scattering of a quantity of interest Y? — PDF f_Y, $\hat{\mu}_Y, \hat{\sigma}_Y$
- What are the parameters that drive the uncertainty on the QoI? — Sensitivity indices
- What is the probability of failure (resp. non performance) of the system? — $p_f = \mathbb{P}\left(Y \geq y_{adm}\right)$
- What is the optimal design (e.g. minimal cost) that guarantees some performance — $\boldsymbol{d}^* = \arg\min \mathfrak{c}(\boldsymbol{d})$ s.t. $\mathbb{P}\left(g(\boldsymbol{X}(\boldsymbol{d}), \boldsymbol{Z}) \leq 0\right) \leq p_{f,adm}$
- What are the best-fit model parameters that allow one to reproduce experimental data — Bayesian inversion

Outline

Why a new UQ software ?

- Huge interest in UQ in the last few years
- (Few) well-established software
  - Covering only a limited spectrum of UQ techniques, *e.g.* sensitivity analysis, Kriging, etc.
  - Difficult to handle as a beginner/non-expert in the field, *e.g.* need for compiling a full platform (+ dependencies)
  - Too complex to use for engineers, *e.g.* built-in scripting language, C/C++ skills needed)
  - Poor interactions with other existing and/or commercial software
  - Non-modular structure requiring highly intrusive development to extend functionality (if possible at all)
  - Limited portability
  - Lack of / limited HPC capabilities
  - Poor/complex documentation

Specifications for UQLab

Based on previous experience, the following requirements have been set:

▪ General purpose UQ software composed of modules that can transparently interact	**General**
▪ Set up / configuration time limited to minutes	**Easy to install ...**
▪ User interface through close-to-natural language (for end users). Learning curve steep (first analysis within one hour)	**... and to use**
▪ Open source scientific algorithms	**Open source**
▪ Identical on windows/linux/mac systems	**Cross-platform**
▪ Thought for high performance computing	**HPC-friendly**
▪ Comprehensive and accurate documentation	**Documented**

Programming language

Requirements

- No compiled language
- High-level pre-existing objects and built-in functions
- Mature environment for easy install
- Well-known in academia AND in the industry

Advantages

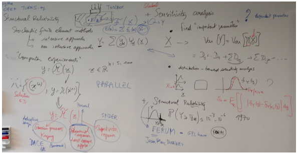
- Quick learning curve
- Standard in academia and industry, well supported
- Object-oriented
- Fast: based on LAPACK libraries
- Portable (Win, MacOS, Linux)
- Powerful debugging-profiling

Drawbacks

- Paying licenses for Matlab (inexpensive for academics)
- HPC support may require additional licenses
- High level libraries not part of basic Matlab

History of development

2011-12	First thoughts on a generic platform suitable for research and dissemination to industry
Nov. 2012	Kick-off meeting



Outcome of the first meeting

Jan.-June 2013	Development of the core (Dr. Stefano Marelli)
2013-2015	Development, documentation and validation of the main modules INPUT (incl.sampling) / PCE / Kriging / Sensitivity analysis
July 1st, 2015	Release of the beta version V0.9
Mar. 1st, 2016	Release of the Structural Reliability module (V0.92)
Oct. 2016	Release of open-source version V1.0

Facts and figures

http://www.uqlab.com

- Release of V0.9 on July 1st, 2015

  ETH license, free of charge for academia

- **450 active licences** in **46 countries**
- About 60% license renewal after one year

Country	**# licences**
United States	83
France	66
Switzerland	55
United Kingdom	24
China	22
Italy	21
India	19
Belgium	16
Germany	14
Brazil	12

UQLab users


Iceland
Finland
Sweden
Norway
Russia
United Kingdom
Poland
Canada
Germany
Ukraine
Kazakhstan
Mongolia
France
Italy
Spain
United States
Turkey
Japan
China
North Atlantic Ocean
South Korea
Afghanistan
Iraq
Iran
Pakistan
Algeria
Libya
Egypt
Mexico
Saudi Arabia
India
Thailand
Mali
Niger
Sudan
Chad
Nigeria
Ethiopia
Venezuela
Colombia
Kenya
DR Congo
Indonesia
Tanzania
Papua New Guinea
Brazil
Peru
Angola
Bolivia
Namibia
Indian Ocean
Madagascar
Botswana
South Atlantic Ocean
Australia
Chile
South Africa
Argentina
Southern Ocean

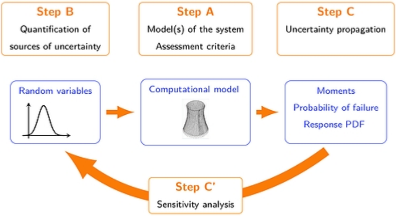
Website

http://www.uqlab.com

UQLab
The Framework for Uncertainty Quantification

OVERVIEW | FEATURES | DOWNLOAD/INSTALL | DOCUMENTATION | EXAMPLES | ABOUT | CONTACT US

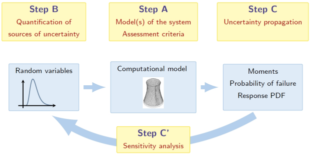
"Make uncertainty quantification available for anybody, in any field of applied science and engineering"

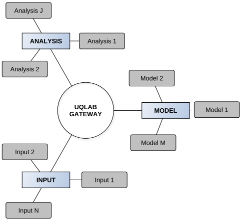


- MATLAB-based Uncertainty Quantification framework
- State-of-the art, highly optimized algorithms
- Easy to use and deploy
- Designed to be extended by users

Outline

UQLab: an implementation of the global UQ framework





An uncertainty quantification problem is defined by:

- A computational model of the physical system
- A probabilistic model of the input uncertainties
- Analysis algorithms

- The Gateway is the entry point/memory management unit
- Core modules (INPUT/ MODEL/ ANALYSIS) keep trace of the objects created by the users (bookkeeping)
- Each object efficiently stores information needed by the underlying algorithms

UQLab: an implementation of the global UQ framework

	Mathematics	UQLab
Computational model	$\boldsymbol{y} = \mathcal{M}(\boldsymbol{x})$	MODEL object
Probabilistic description of input uncertainty	$\boldsymbol{X} \sim f_{\boldsymbol{X}}$	INPUT object
Monte Carlo simulation	$\widehat{\mu}_Y,\ \widehat{\sigma}^2_Y,\ \widehat{f}_Y$	INPUT sampling + MODEL runs
Polynomial chaos expansions	$\sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$	PCE module
Kriging	$\boldsymbol{\beta}^{\mathsf{T}} \cdot \boldsymbol{f}(\boldsymbol{x}) + \sigma^2\, Z(\boldsymbol{x}, \omega)$	KRIGING module
Sensitivity analysis	Importance factors	SENSITIVITY module
Reliability analysis	$P_f = \mathbb{P}\left(\mathcal{M}(\boldsymbol{X}) \leq \bar{m}\right)$	RELIABILITY module

Defining a UQLab problem

- Launch Matlab, start the software by typing `uqlab`
- Create INPUT and MODEL objects by
  - defining their properties
  - using the `uq_createXXX` command

```
% Model
MOpts.mFile = 'myComputationalModel';
myModel = uq_createModel(MOpts); % create MODEL object

% Input random vector
RV.Marginals(1).Type = 'Uniform' ;
RV.Marginals(1).Parameters = [-pi, pi] ;
RV.Marginals(2).Type = 'Lognormal' ;
RV.Marginals(2).Moments = [5, 0.8] ;
...
myInput = uq_createInput(RV);  % create INPUT object
```

- Define analysis options and run it

```
AnOpts.Type = 'Sensitivity';
AnOpts.Method = 'Sobol';
ResObject = uq_createAnalysis(AnOpts); % compute Sobol' indices
```

Storing data in objects

- The commands `uq_createXXX` create objects which store information in a structured way, eg:

`myInput =`

```
uq_input with properties:
         Type: 'uq_default_input'
         Name: 'Input 1'
     Internal: [1x1 struct]
    Marginals: [1x3 struct]
       Copula: [1x1 struct]
     Sampling: [1x1 struct]
      Options: [1x1 struct]
```

`myModel =`

```
uq_model with properties:

            Name: 'Model 1'
    isVectorized: 1
      Parameters: []
           mFile: 'myComputationalModel'
```

Reading / plotting results

- When running an ANALYSIS, the results are stored in a Matlab variable *e.g.* `ResObject`

- Fields of the ANALYSIS outcome can also be read directly:

```
>> ResObject.Results
            Total: [8x1 double]
       FirstOrder: [8x1 double]
    TotalVariance: 832.7455
             Cost: 100000
        ExpDesign: [1x1 struct]
         PCEBased: 0
    VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

- The main features can be printed on screen using `uq_print(ResObject)`

- Context-dependent plots can be obtained using `uq_display(ResObject)`

Bookkeeping of INPUTs and MODELs

- Most analyses require only that a MODEL and an INPUT object have been defined
- For parametric studies, different MODEL / INPUT / ANALYSIS may however coexist in the workspace
- To avoid confusion, the ones used are stored in an `Internal` field of the `ResObject`

```
>> ResObject.Internal
           Type: 'Sensitivity'
         Method: 'sobol'
          Sobol: [1x1 struct]
          Model: [1x1 uq_model]
          Input: [1x1 uq_input]
```

NB: In case several ones have been defined, the **last defined** is the currently used

Bookkeeping of INPUTs and MODELs

- A list of existing INPUT, MODEL and ANALYSIS objects in the workspace can be obtained using uq_listInputs, uq_listModels, uq_listAnalyses
- The active one is tagged with a ">" sign

```
>> uq_listInputs
Available input objects:
  1) Input 1
> 2) Correlated input
```

- The current active INPUT/MODEL/ANALYSIS may be changed by using uq_selectInput/uq_selectModel/uq_selectAnalysis

```
>> uq_selectModel
Available model objects:
  1) R-S model
> 2) R-S model (vectorized)
Please select a model:
```

Outline

Computational models

Analytical functions

```
% String
MOpts.mString = 'X(1) - X(2)'
```

```
% Matlab function handle
MOpts.mHandle = @(X) X(1) - X(2)          % non vectorized
MOpts.mHandle = @(X) X(:,1) - X(:,2)      % vectorized
```

```
% Matlab m-file
MOpts.mFile = 'myComputationalModel'
```

where:

```
function Y = myComputationalModel(X)
  Y = X(:,1) - X(:,2)      % vectorized
```

UQLab user manual – Model module, Report # UQLab-V0.9-103

Computational models

Matlab native code: packaged as a m-file

```
function Y = myQuantityOfInterest(X)
  ...
  ...     % some complex code
  Y = ...;
```

Wrapper of an external code

```
function Y = myQuantityOfInterest(X)
  % pre-process the current input values
    PreProcess(X);
  % write the input file to the third party code
    WriteInputFile;
  % run it from within Matlab
    !ThirdPartyCode.exe theInputFile >   theOutputFile
  % read quantities of interest from the output file
    Y = PostProcess('TheOutputFile');
```

No model: pre-computed data sets

- Surrogate models are built up from experimental designs that may be computed outside UQLab
- Several recent projects have been carried out without any physical link between UQLab and third party codes

Computational models

Precomputed surrogate model

- Polynomial chaos expansion
- Kriging
- *PC-Kriging*
- *Support vector machines*
- *Low-rank tensor approximations*

Outline

Probabilistic models: INPUT module

The uncertain input variables are modelled by a random vector $\boldsymbol{X}$ defined by:

- a set of marginal distributions:
  - 10 parametric distributions incl. Gaussian, lognormal, Gamma, Weibull, etc.
  - Non parametric data-based distributions from kernel smoothing
  - User-defined (supplied by PDF, CDF and inverse CDF functions)
- a copula function for modelling dependence (Gaussian, elliptic)

Reminder: according to Sklar's theorem, the joint cumulative distribution function may be defined through its marginals and copula

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = \mathcal{C}\left(F_{X_1}(x_1), \ldots, F_{X_M}(x_M)\right)$$

where:

- $F_{\boldsymbol{X}}(\boldsymbol{x})$ is the joint CDF
- F_{X_i}'s are marginal distributions
- $\mathcal{C}$ is the copula function

```
% Input random vector
RV.Marginals(1).Type = 'Uniform' ;
RV.Marginals(1).Parameters = [-pi, pi] ;
RV.Marginals(2).Type = 'Lognormal' ;
RV.Marginals(2).Moments = [5, 0.8] ;
...
myInput = uq_createInput(RV);  % create input vector
```

Sampling random vectors

Once an INPUT object is defined, it can be sampled:

```
X1 = uq_getSample(100);           % 100 realizations of current INPUT
X2 = uq_getSample(myInput, 100);  % also specifying the INPUT object
```

Sampling methods in the unit hypercube

- Crude Monte Carlo
- Latin hypercube sampling (+ maximin)
- Quasi-random sequences: Sobol' and Halton points

```
X2 = uq_getSample(200, 'Sobol' )  % 200 Sobol' points from current INPUT
```

Isoprobabilistic transform

- Generalized Nataf transform

Enrichment

- `enrichLHS, enrichSobol, enrichHalton, lhsify` features

UQLab user manual – Input module, Report # UQLab-V0.9-102

Outline

Polynomial chaos expansions

The PCE module allows to build up and use PCE of the form:

$$\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} y_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})$$

Generalized PCE basis

- Classical orthogonal polynomials (Hermite, Legendre, Laguerre, Jacobi)
- Arbitrary polynomials
- Built-in isoprobabilistic transform $\boldsymbol{X} \rightarrow \boldsymbol{U}$ (*e.g.* for lognormal inputs)

Truncation schemes

- Total degree
- Maximum rank
- Hyperbolic q-norm.
- Any combination

UQLab user manual – Polynomial chaos expansions , Report # UQLab-V0.9-104

Polynomial chaos expansions

Coefficients computation

- Projection
  - Full quadrature
  - Smolyak sparse grids
- Ordinary least-squares (a.k.a. regression)
- Sparse PCE by compressive sensing:
  - Least-angle regression (LAR)
  - Orthogonal matching pursuit (OMP)

Output

- PCE (sparse) basis and coefficients
- Moments, LOO error
- Function handle to the PCE surrogate for reuse

Kriging (a.k.a Gaussian process modelling)

The Kriging module allows to build up and use Gaussian process emulators of the form:

$$\tilde{\mathcal{M}}(\boldsymbol{x}) = \boldsymbol{\beta}^{\mathsf{T}} \cdot \boldsymbol{f}(\boldsymbol{x}) + \sigma^2 \, Z(\boldsymbol{x}, \omega)$$

Trend

- Ordinary Kriging (constant trend)
- Universal Kriging (linear, polynomial, PCE trend)
- User-defined (allows for hierarchical Kriging)

Covariance kernels

- Classical 1D kernels: exponential, square-exponential, Matérn + nugget option
- Separable and ellipsoidal multivariate kernels
- User-defined (also non stationary)

UQLab user manual – Kriging , Report # UQLab-V0.9-105

Kriging (a.k.a Gaussian process modelling)

Estimation methods

- Maximum likelihood
- Leave-one-out and leave-K-out cross validation

Optimization algorithms

- Gradient-based (BFGS)
- Genetic algorithm (Matlab ga)
- Hybrid (HGA)

Output

- Mean predictor and Kriging variance (optionally: covariance matrix)
- LOO error

Outline

Sensitivity analysis

Linear(ized) methods

- I/O correlation and rank correlation coefficients
- Standard regression coefficients (SRC)
- Perturbation method (Taylor series expansion)
- Cotter method

Global sensitivity analysis

- Morris method
- Sobol' indices (various MCS estimates)
- PCE-based Sobol' indices

UQLab user manual – Sensitivity analysis , Report # UQLab-V0.9-106

Reliability analysis (a.k.a. rare events estimation)

Approximation methods

- FORM
- SORM

Simulation methods

- Crude Monte Carlo
- Importance sampling
- Subset simulation

Surrogate-based methods

- AK-MCS (active learning based on universal Kriging)
- *PCK-MCS (active learning based on PC-Kriging)*

UQLab user manual – Reliability analysis , Report # UQLab-V0.9-107

High Performance Computing

Objectives

- Provide an interface to common HPC resources
- Local-like execution: no need to manually connect and set-up scheduler scripts and retrieve the results
- Automated book-keeping
- Non-intrusive in user scripts

UQLAB dispatcher module

- Simple text-based credentials
- Supports linux clusters (TORQUE)
- SSH-based implementation
- Non-intrusive
- “Local” feel

```
% Create a dispatcher
DOpts.Profile = 'myCredentials';
DOpts.NCores = 2;
uq_createDispatcher(DOpts);

% Create a model
MOpts.mString = 'X(1)-X(2)';
myModel = uq_createModel(MOpts);

% Evaluate the model
X = [rand(100,1) rand(100,1)];
Y = uq_evalModel(X);
```

Outline

DEMO_01 – Simple Monte Carlo simulation

Problem statement

Model: $\boldsymbol{X} = (R, S)^{\mathsf{T}}\,;\qquad \mathcal{M}(\boldsymbol{X}) = R - S$

Input: $R \sim \mathcal{N}(5,\, 0.8);\qquad S \sim \mathcal{N}(2,\, 0.6)$

Case 1: independent variables

Case 2: linear correlation $\rho_{R,S} = 0.6$

Questions: mean / std.deviation of $R - S$, PDF, $\mathbb{P}\left(M \leq 0\right)$

Solution

	Mean	Std. deviation
No correlation	$5 - 2 = 3$	$\sqrt{0.8^2 + 0.6^2} = 1$
Correlation $\rho_{R,S} = 0.6$	$5 - 2 = 3$	$\sqrt{53/125} = 0.651152$
$\mathbb{P}\left(M \leq 0\right)$	$\Phi(-3) \approx 1.35 \cdot 10^{-3}$	$2.04 \cdot 10^{-6}$

DEMO_02 – Sensitivity analysis

Model: Borehole function

Morris *et al.* (1993)

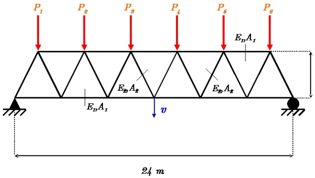
$$\mathcal{M}(\boldsymbol{x}) = \frac{2\pi T_u (H_u - H_l)}{\ln(r/r_w)\left(1 + \frac{2 L T_u}{\ln(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right)}$$

Input

Variable	Distribution	Parameters
Radius of borehole (m) r_w	Gaussian	$\mu = 0.1$
		$\sigma = 0.0161812$
Radius of influence (m) r	Lognormal	$\lambda = 7.71$
		$\zeta = 1.0056$
Transmissivity of upper aquifer (m2/yr) T_u	Uniform	$[63,070\ ;\ 115,600]$
Potentiometric head of upper aquifer (m) H_u	Uniform	$[990\ ;\ 1,110]$
Transmissivity of lower aquifer (m2/yr) T_l	Uniform	$[63.1\ ;\ 116]$
Potentiometric head of lower aquifer (m) H_l	Uniform	$[700\ ;\ 820]$
Length of borehole (m) L	Uniform	$[1,120\ ;\ 1,680]$
Hydraulic conductivity of borehole (m/yr) K_w	Uniform	$[9,855\ ;\ 12,045]$

Questions: Sensitivity indices obtained by different methods

DEMO_03 – Polynomial chaos expansions



10 independent input variables

- 4 describing the bars properties
- 6 describing the loads

Questions

PDF of the max. deflection, statistical moments, probability of failure

$$V = \mathcal{M}^{\mathsf{FE}}\left(E_1, E_2, A_1, A_2, P_1, \dots, P_6\right)$$

Probabilistic model

Parameters	Name	Distribution	Mean	Std. Deviation
Young's modulus	E_1, E_2 (Pa)	Lognormal	2.10×10^{11}	2.10×10^{10}
Hor. bars section	A_1 (m^2)	Lognormal	2.0×10^{-3}	2.0×10^{-4}
Vert. bars section	A_2 (m^2)	Lognormal	1.0×10^{-3}	1.0×10^{-4}
Loads	P_1-P_6 (N)	Gumbel	5.0×10^{4}	7.5×10^{3}

Various PCE constructions

1. Ordinary least-squares: use of an experimental design of size twice the number of PCE coefficients
   - PCE of degree 2
   - PCE of degree 3
   - ...

2. Sparse PCE with LARS, fixed Sobol' points experimental design of size 128, candidate bases of max. degree 2-6

3. Sensitivity analysis (PC-based Sobol' indices)

4. PCE from existing data base

DEMO_04 – Reliability analysis

Limit state function: Hat function

Schöebi et al (2015)

$$g(\boldsymbol{x}) = 10 - (x_1 - x_2)^2 - 8(x_1 + x_2 - 3)^3$$

Input:

$$X_i \sim \mathcal{N}(0,1), \quad i = 1,\, 2$$

Failure probability

$$P_f = \mathbb{P}\left(g(\boldsymbol{X}) \leq 0\right)$$

Reliability methods

- Crude Monte Carlo simulation
- FORM
- Importance sampling
- Subset simulation
- AK-MCS

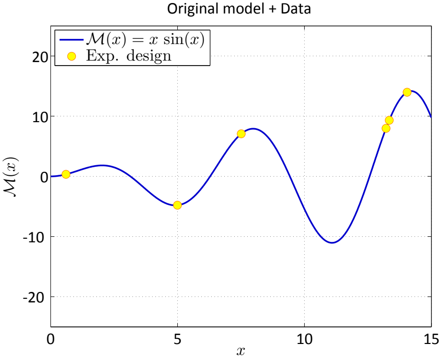
DEMO_05 – Kriging (1D)

Computational model

$$x \mapsto x \sin x \qquad \text{for} \quad x \in [0,\, 15]$$

Experimental design

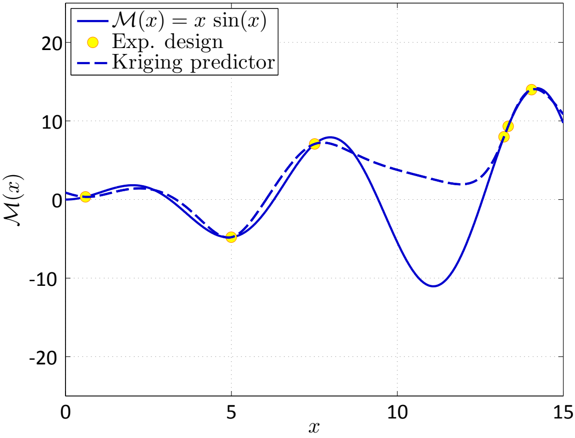
Six points selected in the range $[0,\, 15]$ using Monte Carlo simulation:



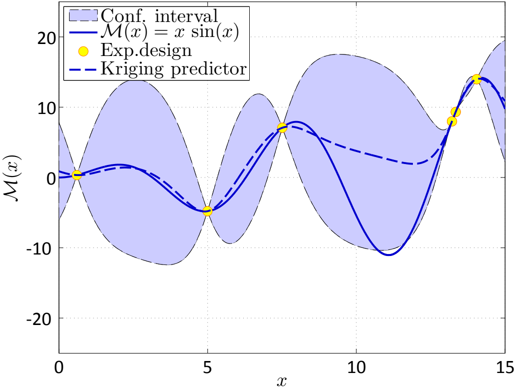
$$\mathcal{X} = \{0.6042 \quad 4.9958 \quad 7.5107 \quad 13.2154 \quad 13.3407 \quad 14.0439\}$$

Kriging predictor

```
Trend.Type = 'ordinary' ;              % Ordinary Kriging
Covariance.Type = 'matern-5_2';        % Matérn 5/2
EstimMethod = 'ML';                    % Maximum likelihood
Optim.Method = 'BFGS';                 % BFGS algorithm
```

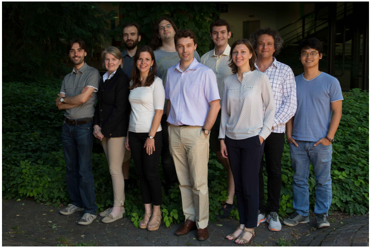


Confidence bounds



- The Kriging variance gives a local error estimator about the accuracy of the predictor (NB: the variance is equal to zero at points of the experimental design)
- Large confidence intervals correspond to a large (epistemic) uncertainty on the prediction

Questions ?



Chair of Risk, Safety & Uncertainty Quantification

www.rsuq.ethz.ch

The Uncertainty Quantification Laboratory

www.uqlab.com

Thank you very much for your attention !