Treatment of uncertainties in multi-physics model for wind turbine asset management

PhD first year overview

E. Fekhari [1] B. Iooss [1 2] V. Chabridon [1] M. Capaldo [1]

[1]EDF R&D - 6 quai Watier, Chatou, France

[2]Université Nice Côte d'Azur - 28 Avenue de Valrose, Nice, France

September 16, 2021

Industrial context
Numerical simulation code
Mean damage estimation
Analytical example
Conclusion

Industrial context

- EDF Renewables $\sim$ 10 000 MW of wind turbine (WT) worldwide
- Deterministic design of WT for 25 year lifetime
- In the future:
  - Wind farms reaching end-of-life
  - New technology: offshore floating WT

Needs probabilistic tools to optimize safety margins and asset management

Industrial context

- EDF Renewables $\sim$ 10 000 MW of wind turbine (WT) worldwide
- Deterministic design of WT for 25 year lifetime
- In the future:
  - ▷ Wind farms reaching end-of-life
  - ▷ New technology: offshore floating WT

Needs probabilistic tools to optimize safety margins and asset management

Uncertainty quantification (UQ) for:

- Fine power production estimation
- Safe lifetime extension regarding tower damage
- Probabilistic design for floating WT (multiple technologies)

WT offshore technologies

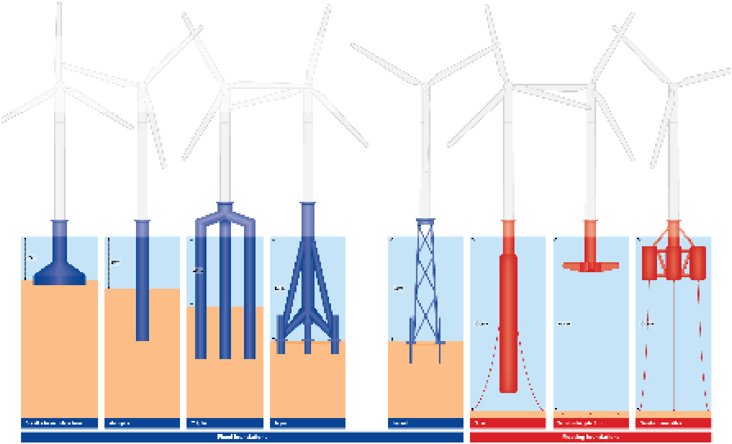


Figure 1: Wind turbine offshore technologies[1]

[1] https://www.windpowermonthly.com/article/1210054

WT studied failure mode

Studied failure mode: mechanical damage in tower welded joints

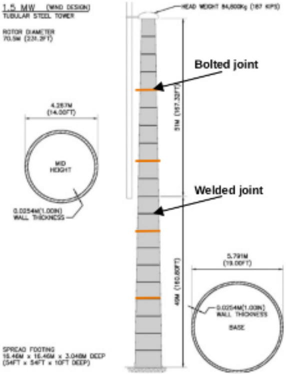


Figure 2: Illustrative tower assembly[1]

[1] M. LaNier. LWST Phase I Project Conceptual Design Study: Evaluation of Design and Construction Approaches for Economical Hybrid Steel/Concrete Wind Turbine Towers. 2005.

TurbSim: turbulent wind field simulation

TurbSim is a stochastic, full-field, turbulence simulator (NREL[1])

- **inputs**: mean wind, wind direction, wind shear, turbulence model, turbulence intensity, hub height, simulation time, etc.
- **outputs**: wind speed field

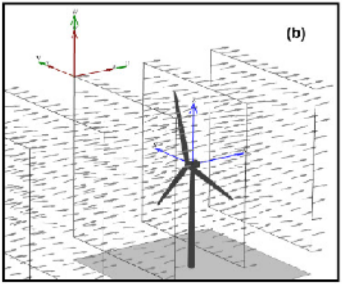


Figure 3: Illustrative wind speed field simulated

[1] B. Jonkman. *Turbsim User's Guide: Version 1.50*. 2009.

DIEGO: Hydro-Aero-Servo-Elasto simulation

Dynamique Intégrée des Eoliennes et Génératrices Offshore, DIEGO is WT multi-physics simulator (EDF R&D[2])

- **inputs**: TurbSim's output, WT geometry, material properties, soil stiffness, controller properties.
- **outputs**: power production, mechanical stress, displacements, etc.

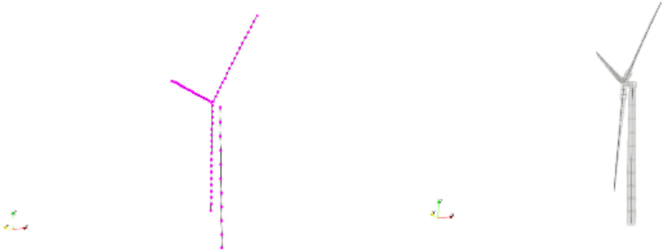


Figure 4: Illustrative structural and aero-dynamic mesh from DIEGO

[2] M. Capaldo et al. "Influence of cracks on the buckling of wind turbine towers". In: *Journal of Physics: Conference Series* (2020).

Mechanical damage assessment

The damage is computed at specific points of the structure

1. Equivalent Von Mises stress time series
2. Fatigue stress cycles identification using Rainflow counting method[3]
3. Damage computation using Miner's rule[4]

CPU time: up to 20 min per simulation of the chain

[3] DNV-GL. *DNVGL-RP-C203: Fatigue design of offshore steel structures*. Tech. rep. 2016.

[4] D. Wilkie and C. Galasso. "Gaussian process regression for fatigue reliability analysis of offshore wind turbines". In: *Structural Safety* (2021).

Mechanical damage assessment

The damage is computed at specific points of the structure

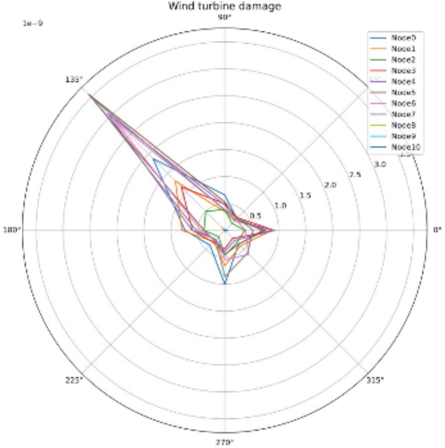


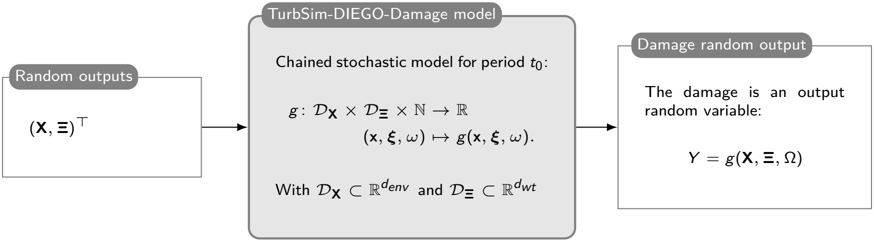
Figure 5: Damage computation of WT tower

Chain simulation codes

Wrap the TurbSim – DIEGO – Damage chain of simulation codes

- Python distributed wrapper (using the `otwrapy` module based on OpenTURNS[5])
- Deployed on EDF R&D High Performance Computers facility

TurbSim is a stochastic model controlled by a pseudo-random seed ω



$\mathbf{X}$ environmental random vector characterized by its joint distribution $f_{\mathbf{X}}(\cdot)$

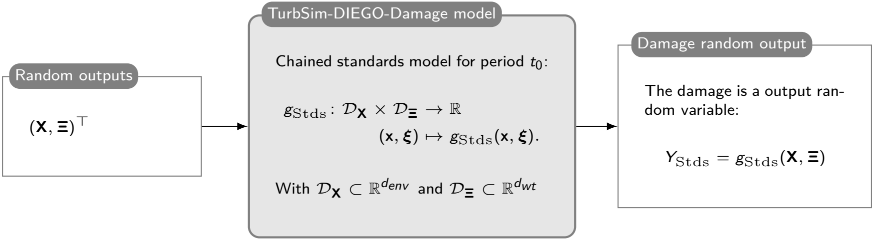
$\mathbf{\Xi}$ wind tubine random vector characterized by its joint distribution $f_{\mathbf{\Xi}}(\cdot)$

[5] `http://openturns.github.io/otwrapy/master/`

Chain simulation codes

Standards[6] recommend to repeat each simulation for $n_{rep} \in \mathbb{N}$ pseudo-random seeds and average the outputs

$$g_{\mathrm{Stds}}(\mathbf{x}, \boldsymbol{\xi}) := \frac{1}{n_{rep}} \sum_{i=1}^{n_{rep}} g(\mathbf{x}, \boldsymbol{\xi}, \omega^{(i)})$$



$\mathbf{X}$ continuous random vector characterized by its joint distribution $f_{\mathbf{X}}(\cdot)$

$\boldsymbol{\Xi}$ wind tubine random vector characterized by its joint distribution $f_{\boldsymbol{\Xi}}(\cdot)$

[6] DNV-GL. *DNVGL-ST-0437: Loads and site conditions for wind turbines*. Tech. rep. 2016.

Output quantity of interest: Damage Equivalent Load

Considering the wind turbine random vector as known $\boldsymbol{\Xi} = \boldsymbol{\xi}$

Standards Damage Equivalent Load (DEL)

Damage Equivalent Load is the expected value of the damage over the environmental conditions (random vector $\mathbf{X}$)

$$\mathbb{E}_{\mathbf{X}}[Y_{\mathrm{Stds}}] = \mathbb{E}_{\mathbf{X}}[g_{\mathrm{Stds}}(\mathbf{X})] = \int_{\mathbf{X}} g_{\mathrm{Stds}}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x}$$

Fully stochastic Damage Equivalent Load (DEL)

Since $(\Omega, \mathbf{X})$ are independent, the previous integral can generalized:

$$\begin{aligned}\mathbb{E}_{\mathbf{X},\Omega}[Y] = \mathbb{E}_{\mathbf{X},\Omega}[g(\mathbf{X},\Omega)] &= \iint_{\mathbf{X},\Omega} g(\mathbf{x},\omega) f_{\mathbf{X},\Omega}(\mathbf{x},\omega)\, d\mathbf{x}\, d\omega \\ &= \iint_{\mathbf{X},\Omega} g(\mathbf{x},\omega) f_{\mathbf{X}}(\mathbf{x}) f_{\Omega}(\omega)\, d\mathbf{x}\, d\omega\end{aligned}$$

Methods for DEL estimation

- Sampling methods for numerical integration
  - Monte Carlo
  - Low discrepancy sequences[7]
  - Latin Hypercube Sampling
- Metamodel + sampling method for numerical integration
  - Space-Filling learning set (one shot)[8]
  - Adaptive learning set, e.g., AKDA (iterative)[9]
- Quadrature methods for numerical integration[10]

---

[7] K. Müller and P. Cheng. "Application of a Monte Carlo procedure for probabilistic fatigue design of floating offshore wind turbines". In: *Wind Energy Science* (2018).

[8] R. Slot et al. "Surrogate Model Uncertainty in Wind Turbine Reliability Assessment". In: *Renewable Energy* (2019).

[9] Q. Huchet. "Kriging based methods for the structural damage assessment of offshore wind turbines". PhD thesis. 2019.

[10] L. van den Bos. "Quadrature Methods for Wind Turbine Load Calculations". PhD thesis. 2020.

Industrial context
Numerical simulation code
Mean damage estimation
Analytical example
Conclusion

Toy case: Branin-Hoo function

Branin-Hoo function

$$r(x_1, x_2) = h \circ t(x_1, x_2)$$
$$t(x_1, x_2) = ((15x_1 - 5), (15x_2))^\top$$
$$h(u_1, u_2) = \frac{\left(u_2 - 5.1\frac{u_1^2}{4\pi^2} + 5\frac{u_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(u_1) + 10 - 54.8104}{51.9496}$$

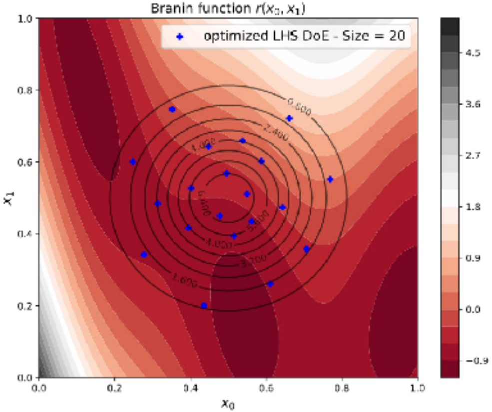
Random inputs:

$$X_1 \sim \mathcal{N}(\mu, \sigma);\ \ X_2 \sim \mathcal{N}(\mu, \sigma)$$
$$\mathbf{X} = (X_1, X_2)^\top$$

We estimate the central tendency of $Y = r(\mathbf{X})$:

$$\mathbb{E}[Y] = \int_{\mathcal{D}_X} r(x) f_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x}$$

Toy case: Branin-Hoo function



- The DoE is not built on a uniform distribution since our objective is not a globally-accurate surrogate modeling
- The LHS DoE is optimized using the Simulated Annealing algorithm with the L^2-centered discrepancy as space-filling criterion

Validation procedure of the mean

Compute a reference mean $m_{ref} = \widehat{Y}$ on a large Monte Carlo ($N_{ref} = 10^9$) using the "true function"

```
Mean = -0.36451364108542894
Standard deviation = 1.5434486434042638e-05
Number of calls to the g function = 1000000000
Coef. of var.= 0.000042
Monte Carlo 95% IC: [-0.3645136420420683, -0.36451364012878956]
```

Validation procedure of the mean

Compute a reference mean $m_{ref} = \widehat{Y}$ on a large Monte Carlo ($N_{ref} = 10^9$) using the "true function"

```
Mean = -0.36451364108542894
Standard deviation = 1.5434486434042638e-05
Number of calls to the g function = 1000000000
Coef. of var.= 0.000042
Monte Carlo 95% IC: [-0.3645136420420683, -0.36451364012878956]
```

Compare the performances of mean estimation methods

1. Build a Monte Carlo validation sample $\mathcal{X}_{mc} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}), N = 10^7$
2. Compute an approximated mean $m_{appx} = \widehat{Y}_{kri}$ on $\mathcal{X}_{mc}$ using a Kriging model
3. Compute number of correct digits between m_{ref} and m_{appx} :

$$N(m_{ref}, m_{appx}) = \log_{10} \left| \frac{m_{ref}}{m_{ref} - m_{appx}} \right| .$$

AK-DA: "Adaptive Kriging Damage Assessment"

Algorithm 1: AK-DA

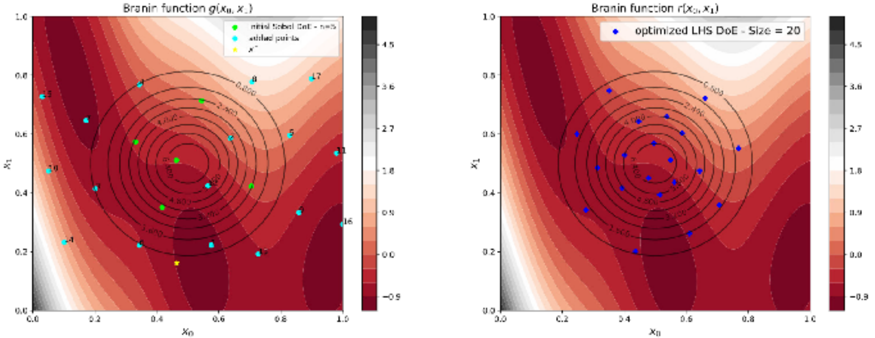
Build a fine regular grid $\mathcal{L}_{grid}$
Compute $f_{\mathbf{X}}(\mathbf{x}^{(k)})$ for $\mathbf{x}^{(k)} \in \mathcal{L}_{grid}$
Build an initial learning set $\mathcal{X}_n = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}) \in \mathcal{D}_X$
while *condition* **do**
  Build a Kriging model on the learning set
  Compute Kriging variance $s_n^2(\mathbf{x}^{(k)})$ for $\mathbf{x}^{(k)} \in \mathcal{L}_{grid}$
  Compute acquisition function $\mathcal{A}_n(\mathbf{x}^{(k)}) = s_n^2(\mathbf{x}^{(k)}) \cdot f_{\mathbf{X}}(\mathbf{x}^{(k)})$
  Find

$$\mathbf{x}^*_{n+1} = \underset{\mathbf{x}^{(i)} \in \mathcal{L}_{grid}}{\arg\max}\, \mathcal{A}_n(\mathbf{x}^{(i)})$$

  Add this point to the learning set, $\mathcal{X}_{n+1} = \{\mathcal{X}_n, \mathbf{x}^*_{n+1}\}$
end

Note that the optimization could be done without regular grid. Probably not a big added value in small dimension.

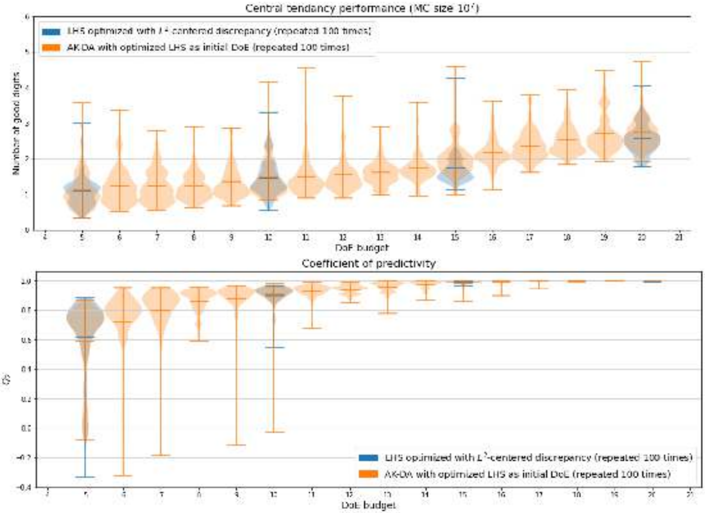
AK-DA vs. Optimized LHS

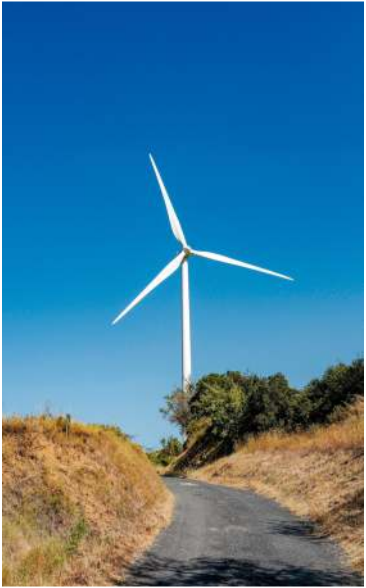


- For this function, AK-DA tends to explore the space more than LHS
- How to build the initial DoE?
  - ▷ dimension and function dependent
  - ▷ Kriging variance needs to be validated for the acquisition function ⇒ initial DoE should be iterative (low disc. sequences, FSSF, Kernel Herding)

AK-DA vs Optimized LHS

LHS optimized on the input random vector (repeated 100 times)





Conclusions

- Toy-function very different from the industrial case (complexity, dimension)
- Here, AKDA performs similarly with Kriging on optimized LHS
- AKDA improved by validating Kriging variance before applying acquisition function
- Need an automated benchmark for central tendency estimation (using `otbenchmark` module[11])

[11] E. Fekhari et al. "otbenchmark: an open source python package for benchmarking and validating uncertainty quantification algorithms". In: *Proc. of the 4th International Conference on Uncertainty Quantification in Computational Sciences and Engineering*. 2021.

Perspectives

Ongoing work

- Contribution to EU-founded project HIPERWIND[12] (DTU, IFPEN, ETH Zürich)
- Iterative design of experiments for metamodel validation

Perspectives

- Industrial use-case of Teesside offshore wind farm
  - ▷ UQ of the environmental random variables (dependent measures available)
  - ▷ Stochastic metamodeling of the damage without replications
  - ▷ Benchmark methods to estimate DEL
  - ▷ Screen non-influential variables among wind turbine variables
  - ▷ Reliability analysis of the DEL over wind turbine variables (e.g., material properties)
- Reliability-based design optimization (RBDO) of floating WT model

[12] https://www.hiperwind.eu/

Bibliography

F. Bachoc. “Estimation paramétrique de la fonction de covariance dans le modèle de Krigeage par processus Gaussiens. Application à la quantifcation des incertitudes en simulation numérique”. PhD thesis. 2013.

L. van den Bos. “Quadrature Methods for Wind Turbine Load Calculations”. PhD thesis. 2020.

M. Capaldo et al. “Influence of cracks on the buckling of wind turbine towers”. In: *Journal of Physics: Conference Series* (2020).

DNV-GL. *DNVGL-RP-C203: Fatigue design of offshore steel structures*. Tech. rep. 2016.

DNV-GL. *DNVGL-ST-0437: Loads and site conditions for wind turbines*. Tech. rep. 2016.

E. Fekhari et al. “otbenchmark: an open source python package for benchmarking and validating uncertainty quantification algorithms”. In: *Proc. of the 4th International Conference on Uncertainty Quantification in Computational Sciences and Engineering*. 2021.

Q. Huchet. “Kriging based methods for the structural damage assessment of offshore wind turbines”. PhD thesis. 2019.

B. Jonkman. *Turbsim User’s Guide: Version 1.50*. 2009.

K. Müller and P. Cheng. “Application of a Monte Carlo procedure for probabilistic fatigue design of floating offshore wind turbines”. In: *Wind Energy Science* (2018).

B. Shang and D. Apley. “Fully-sequential space-filling design algorithms for computer experiments”. In: *Journal of Quality Technology* (2020).

R. Slot et al. “Surrogate Model Uncertainty in Wind Turbine Reliability Assessment”. In: *Renewable Energy* (2019).

D. Wilkie and C. Galasso. “Gaussian process regression for fatigue reliability analysis of offshore wind turbines”. In: *Structural Safety* (2021).

Monte Carlo estimation of Y - Deterministic model

Empirical mean estimator

Let $\left\{\mathbf{X}^{(i)}\right\}_{i=1,\dots,n}$ for $n \in \mathbb{R}$, a i.i.d sample of the random vector $\mathbf{X}$. The empirical mean estimator can be written:

$$\widehat{Y} = \frac{1}{n}\sum_{i=1}^{n} d\left(\mathbf{X}^{(i)}\right).$$

We can demonstrate that $\mathbb{E}[\widehat{Y}] = \mathbb{E}[Y] = \mu_d$ and $Var(\widehat{Y}) = \frac{Var(Y)}{n} = \frac{\sigma_d m^2}{n}$ therefore $\delta_{\widehat{Y}} = \frac{\sqrt{Var(Y)}}{\sqrt{n}\cdot\mu_d}$.

Central limit theorem

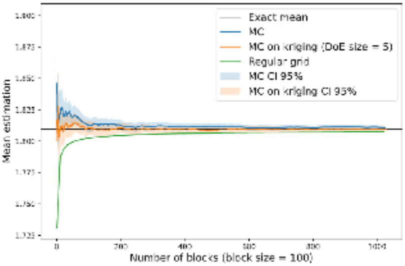
$$\widehat{Y} \xrightarrow{d} \mathcal{N}\left(\mu_d, \frac{\sigma_d^2}{n}\right)$$

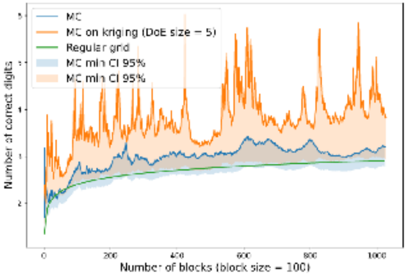
Central tendency estimation: regular grid *vs.* Monte Carlo

Regular grid drawbacks:

- No full control of the central tendency convergence
- Convergence is slower! Note that Quasi-MC was not used yet.

Illustration on a simple function (BBRC RP22[13]) using uniform inputs:





[13] https://rprepo.readthedocs.io/en/latest/

Simple Kriging

1. Assuming that $d(\mathbf{x})$ is a realization of the gaussian process (GP) $D(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$.
2. Considering the learning DoE: $\begin{cases} \mathcal{X}_{obs} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}) \\ \mathbf{d_{obs}} = \left[d(\mathbf{x}^{(1)}), \dots, d(\mathbf{x}^{(n)})\right]^\top \end{cases}$
3. Therefore, $\mathbf{D_{obs}} = \left[D(\mathbf{x}^{(1)}), \dots, D(\mathbf{x}^{(n)})\right]^\top$ is a gaussian vector, and one can write the following joint distribution: $\begin{pmatrix} D(\mathbf{x}) \\ \mathbf{D_{obs}} \end{pmatrix}$.
4. The Kriging model $\widetilde{D}(\mathbf{x})$ is the GP $D(\cdot)$ conditioned by the observations $\mathbf{d_{obs}}$

$$\widetilde{D}(\mathbf{x}) = [D(\mathbf{x}) | \mathbf{D_{obs}} = \mathbf{d_{obs}}, \beta, \sigma, \boldsymbol{\theta}] \sim \mathcal{N}(\widetilde{d}(\mathbf{x}), s(\mathbf{x})^2).$$

5. With $\{\beta, \sigma, \boldsymbol{\theta}\}$ called hyper-parameters, usually estimated by MLE.
6. With $\widetilde{d}(\mathbf{x})$ called Kriging predictor and $s(\mathbf{x})^2$ called Mean Squared Error, both defined analytically.

OpenTURNS implementation (1/4)

Problem definition

```
# Import the packages
import openturns as ot
# Define Branin-Hoo function
branin = ot.SymbolicFunction(['x1', 'x2'],
      ['((x2-(5.1/(4*pi_^2))*x1^2+5*x1/pi_-6)^2+10*(1-1/(8*pi_))*cos(x1)+10-54.8104)
     /51.9496'])
transfo = ot.SymbolicFunction(['u1', 'u2'], ['15*u1-5', '15*u2'])
g = ot.ComposedFunction(branin, transfo)
# Create a random joint distribution
trunc_normal = ot.TruncatedDistribution(ot.Normal(0.5, 0.15), 0., 1.)
distribution = ot.ComposedDistribution([trunc_normal, trunc_normal])
```

Learning and validation set

```
# Build a LHS Experiment: learning set
experiment = ot.LHSExperiment(distribution, 10)
in_learn_sample = experiment.generate()
out_learn_sample = g(in_learn_sample)
# Build a Sobol Sequence: test set
sequence = ot.SobolSequence(2)
experiment = ot.LowDiscrepancyExperiment(sequence, distribution, 128)
in_test_sample = experiment.generate()
out_test_sample = g(in_test_sample)
```

OpenTURNS implementation (3/4)

Build a Kriging model (Default MLE solver failed -> multi-start method applied)

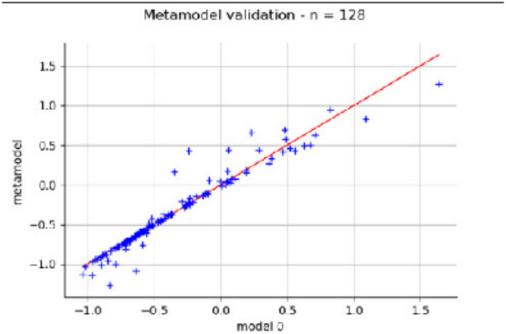
```
# Define a Squared Exponential kernel (Scale=[1, 1], amplitude=1)
covarianceModel = ot.SquaredExponential([1., 1.], [1.])
# Build a constant function basis
basis = ot.ConstantBasisFactory(dim).build()
# Build default Kirging model
kriging = ot.KrigingAlgorithm(in_learn_sample, out_learn_sample, covarianceModel, basis)
kriging.run()
```

Results and validation

```
# Get kringing results and metamodel
krigingResult = kriging.getResult()
kriging_mean = krigingResult.getMetaModel()
# Compute Q2 on the Sobol test set
val = ot.MetaModelValidation(in_test_sample, out_test_sample, kriging_mean)
q2 = val.computePredictivityFactor()
# Prints
print("Optimal scale = ", krigingResult.getCovarianceModel().getScale())
print("Optimal amplitude = ", krigingResult.getCovarianceModel().getAmplitude())
print("Optimal trend coefficients = ", krigingResult.getTrendCoefficients())
print("Q2 = ",q2)
```

```
Optimal scale= [0.218602,0.474411]
Optimal amplitude = [1.0884]
Optimal trend coefficients = [[-0.0938758]]
Q2 =  0.928937137828958
```

OpenTURNS implementation (4/4)

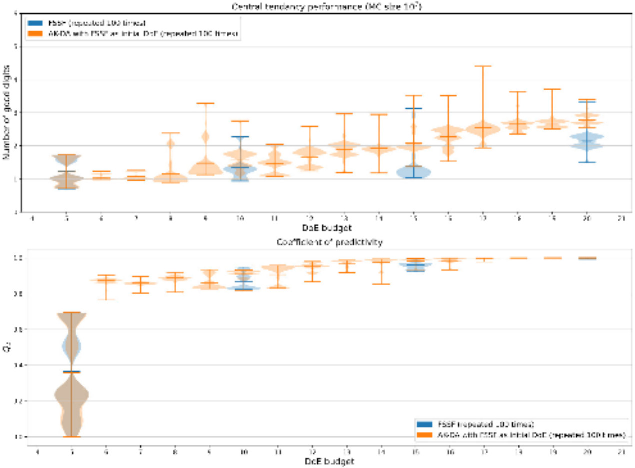


OK but how good is the mean estimation?

⇒Metamodel for a QoI

FSSF

Fully Sequential Space Filling (FSSF[14]) input DoE (repeated 100 times)



[14] B. Shang and D. Apley. "Fully-sequential space-filling design algorithms for computer experiments". In: *Journal of Quality Technology* (2020).

Improved AK-DA

Idea inspired by Bayesian optimization:

- Iteratively validate the Kriging variance using the Predictive Variance Adequation[15]
- Then start the adaptive enrichment strategy (AKDA)

Let be the test set $\mathcal{X}_p = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(p)}) \subset \mathcal{D}_X$, disjoint from the learning set $\mathcal{X}_n$,

$$PVA := \left| \log_{10} \left(\frac{1}{p} \sum_{i=1}^{p} \frac{(d(\mathbf{x}^{(i)}) - \widetilde{d}(\mathbf{x}^{(i)}))^2}{s(\mathbf{x}^{(i)})^2} \right) \right| .$$

In practice: Start AK-DA once $Q_2 > 0.7$ and $PVA < 2$

[15] F. Bachoc. "Estimation paramétrique de la fonction de covariance dans le modèle de Krigeage par processus Gaussiens. Application à la quantifcation des incertitudes en simulation numérique". PhD thesis. 2013.