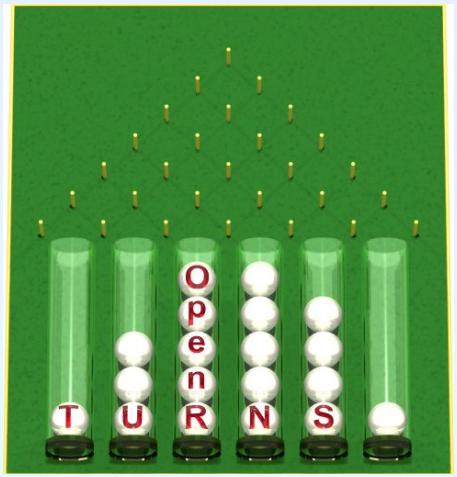
# Chaos polynomial dans Open TURNS

Régis LEBRUN

EADS Innovation Works




regis.lebrun@eads.net

22 juin 2009

# Plan de l'exposé



# Objectifs

## Représenter une variable aléatoire

Soient $\underline{X} \hookrightarrow D_X$ un vecteur aléatoire de $d$ paramètres incertains, $f \in L^2_{D_X}(\mathbb{R}^d, \mathbb{R})$ un modèle numérique, $Y = f(\underline{X})$ une grandeur d'intérêt.
On introduit $\underline{Z} \hookrightarrow \mu_Z$ tel que $\dim \underline{Z} = \dim \underline{X}$.
On cherche à représenter $Y$ à l'aide de $\underline{Z}$ et d'une série de fonctions orthonormées formant une famille complète par rapport à $\mu_Z$ :

$$Y = g(\underline{Z}) = \sum_{k=1}^{\infty} \alpha_k \Phi_k(\underline{Z}) \text{ avec } \mathbb{E}[\Phi_i(\underline{Z})\Phi_j(\underline{Z})] = \delta_{ij} \quad (1)$$

et après troncature on obtient :

$$\tilde{Y} = \sum_{k \in K} \alpha_k \Phi_k(\underline{Z}) \text{ avec } \mathrm{card}(K) < \infty \text{ et } \mathbb{E}[(Y - \tilde{Y})^2] \ll \mathbb{E}[Y^2] \quad (2)$$

# Objectifs

## Approcher une fonction numérique

De (1) on déduit, si $T$ est une transformation isoprobabiliste entre $\underline{X}$ et $\underline{Z}$ telle que $\underline{Z} = T(\underline{X})$ :

$$Y = f(\underline{X}) = g(T(\underline{X})) = \sum_{k=1}^{\infty} \alpha_k \Phi_k(T(\underline{X})) \tag{3}$$

d'où un modèle approché $\tilde{f}$ pour $f$ en utilisant (2) :

$$f(\underline{x}) \simeq \tilde{f}(\underline{x}) = \sum_{k \in K} \alpha_k \Phi_k(T(\underline{x})) \tag{4}$$

# Etape 1

## Choix de $\underline{Z}$ et construction de $(\Phi_k)_{k\in\mathbb{N}}$

- Si $\underline{Z}$ est à composantes indépendantes, $\mu_Z = \prod_{j=1}^{d} \mu_{Z_j}$
- Si $(\phi_n^j)_{n\in\mathbb{N}}$ est une famille complète de fonctions orthonormées par rapport à $\mu_{Z_j}$, la famille $(\Phi_k)_{k\in\mathbb{N}}$ est complète et orthonormée par rapport à $\mu_Z$, avec :

$$\Phi_k(\underline{z}) = \prod_{j=1}^{d} \phi_{n_j}^j(z_j) \tag{5}$$

où $(n_1, \ldots, n_d) = \tau(k)$, $\tau$ bijection de $\mathbb{N}$ dans $\mathbb{N}^d$

- Si $\mu_{Z_j}$ est telle que $\mathbb{E}[|Z_j^m|] < \infty$ pour tout $m$, alors on peut prendre la famille $(\phi_n^j)_{n\in\mathbb{N}}$ polynômiale. La famille $(\Phi_k)_{k\in\mathbb{N}}$ est alors polynômiale, c'est la base du chaos polynômial.

# Etape 2

## Choix de $\tau$ et de $K$

La fonction d'énumération $\tau$ décrit la manière d'explorer la famille complète $(\Phi_k)_{k\in\mathbb{N}}$, et l'ensemble $K$ le nombre fini d'éléments de cette famille retenus pour l'approximation par chaos polynômial. Cette sélection correspond à une stratégie adaptative permettant de maîtriser la qualité de l'approximation dans (2).

- Un choix naturel pour $\tau$ est l'indexation lexicographique par degré total croissant :
  - $\tau(0) = (0, \ldots, 0)$
  - si $m < n$, soit $\sum_{j=1}^{d} \tau_j(m) < \sum_{j=1}^{d} \tau_j(n)$, soit $\exists j^* \leq d$ tel que $\forall j < j^*, \tau_j(m) = \tau_j(n)$ et $\tau_{j^*}(m) < \tau_{j^*}(n)$.
- Un choix usuel pour $K$ est de prendre $K = \{0, \ldots, N-1\}$ avec $N = \mathrm{card}\{k, \sum_{j=1}^{d} \tau(k) \leq D\} = \frac{(d+D)!}{d!D!}$ où $D$ est un degré total donné.

# Etape 3

## Calcul des $\alpha_k$

C'est l'étape de projection de $g$ sur $(\Phi_k)_{k \in K}$. La famille étant orthonormée par rapport à $\mu_Z$, on a :

$$(\alpha_k)_{k \in K} = \underset{(\tilde{\alpha}_k)_{k \in K}}{\operatorname{argmin}} \mathbb{E}\left[\left(g(\underline{Z}) - \sum_{k=1}^{N} \tilde{\alpha}_k \Phi_k(\underline{Z})\right)^2\right] \quad (6)$$

$$\Longleftrightarrow (\alpha_k)_{k \in K} = (\mathbb{E}[g(\underline{Z})\Phi_k(\underline{Z})])_{k \in K} \quad (7)$$

La relation (6) correspond à un problème d'approximation par moindres carrés, la relation (7) à un calcul de produits scalaires : le choix de la méthode de calcul correspond à une stratégie de projection.

# Cas Ishigami

## Présentation du cas

On a $\underline{X} = (X_1, X_2, X_3)$ avec $X_i \hookrightarrow \mathcal{U}(-\pi, \pi)$ et les $X_i$ indépendantes. Le modèle s'écrit $(a = 7, b = 1/10)$ pour les tests) :

$$f(x_1, x_2, x_3) = \sin(x_1) + a \sin^2 x_2 + b x_3^4 \sin x_1 \tag{8}$$

On a un budget calcul de 500 évaluations du modèle, on souhaite obtenir la moyenne, la variance, les indices de Sobol du premier ordre de $Y$, ainsi qu'un modèle approché de $f$. Dans OpenTURNS on écrit :

```
1  a = 7.0
2  b = 0.1
3  inputVariables = Description()
4  inputVariables.add("x1")
5  inputVariables.add("x2")
6  inputVariables.add("x3")
...
```

# Cas Ishigami

## Présentation du cas

```
 7  outputVariables = Description()
 8  outputVariables.add("y")
 9  formula = Description()
10  formula.add("sin(x1) + ("+str(a)+") * (sin(x2))^2 + ("+
    str(b)+") * x3^4 * sin(x1)")
11  model = NumericalMathFunction(inputVariables,
    outputVariables, formula)
12  dimension = inputVariables.getSize()
13  marginals = DistributionCollection(dimension,
    Distribution(Uniform(-pi, pi)))
14  distribution = Distribution(ComposedDistribution(
    marginals))
...
```

# Création de la base

## Choix des $(\phi_k^j)_{k\in\mathbb{N}}$

Dans Open TURNS, la distribution $\mu_Z$ est construite implicitement à partir de la construction des $(\phi_k^j)_{k\in\mathbb{N}}$. Ce sont des OrthogonalUniVariatePolynomialFamily particulières, ici des LegendreFactory. La distribution $\mu_{Z_j}$ sera alors $\mathcal{U}(0,1)$. Dans OpenTURNS on écrit :

```
15  polynomialCollection = PolynomialFamilyCollection(
    dimension, OrthogonalUniVariatePolynomialFamily(
    LegendreFactory()))
...
```

# Création de la base

## Choix de $\tau$ et création des $(\Phi_k)_{k\in\mathbb{N}}$

La famille $(\Phi_k)_{k\in\mathbb{N}}$ est obtenue par tensorisation (5), une OrthogonalProductPolynomialFactory. Il faut également choisir la fonction $\tau$ qui sera une EnumerateFunction. Dans OpenTURNS on écrit :

```
16  enumerateFunction = EnumerateFunction(dimension)
17  productBasis = OrthogonalBasis(OrthogonalProductPolynomial
    Factory(polynomialCollection, enumerateFunction))
...
```

# Stratégie de sélection

## Construction de $K$

La construction de $K$ se fait par le choix d'une AdaptiveStrategy particulière. La sélection peut être faite sur un critère à priori, comme le degré total maximal des $\Phi_k$ (la FixedStrategy), ou sur des critères plus sophistiqués basés sur la contribution des $\Phi_k$ à la norme $L^2_{\mu_{\mathbf{Z}}}$ de $\tilde{Y}$ (la CleaningStrategy). Dans OpenTURNS, on écrit :

```
18  totalDegree = 9
19  maxSize = enumerateFunction.getStrateCumulated
    Cardinal(totalDegree)
20  adaptiveStrategy = AdaptiveStrategy(FixedStrategy(
    productBasis, maxSize))
...
```

# Stratégie de projection

## Calcul des $\alpha_k$

OpenTURNS propose de calculer les $\alpha_k$ par moindres carrés (LeastSquaresStrategy). Différents plans d'expériences (WeightedExperiments) peuvent être utilisés pour discrétiser l'espérance de la formulation (6). Dans OpenTURNS, on écrit :

```
21  samplingSize = 500
22  projectionStrategy = ProjectionStrategy(
    LeastSquaresStrategy(LHSExperiment(samplingSize)))
...
```

# Résolution globale

## Calcul de l'approximation

L'algorithme FunctionalChaosAlgorithm est maintenant près à être lancé. Le résultat est accessible dans un objet FunctionalChaosResult. Dans OpenTURNS, on écrit :

```
23  algo = FunctionalChaosAlgorithm(model, distribution,
    adaptiveStrategy, projectionStrategy)
24  algo.run()
25  result = algo.getResult()
...
```

# Exploitation

## Cœfficients, méta-modèle, moments, indices de Sobol etc.

L'objet FunctionalChaosResult donne directement accès aux $\alpha_k$, à $K$, à $\tau$, à $T$ et aux $\Phi_k$, les grandeurs statistiques sont accessibles via un FunctionalChaosRandomVector construit à partir d'un FunctionalChaosResult. Dans OpenTURNS, on écrit :

```
26  print "coefficients =", result.getCoefficients()
27  print "meta-model =", result.getMetaModel()
28  vector = FunctionalChaosRandomVector(result)
29  print "mean =", vector.getMean()[0]
30  print "variance =", vector.getVariance()[0, 0]
31  for i in range(dimension) :
32     print "Sobol index", i, "=", vector.getSobolIndice(i)
```

En 32 lignes, tout compris !

# Valeurs numériques (FixedStrategy)

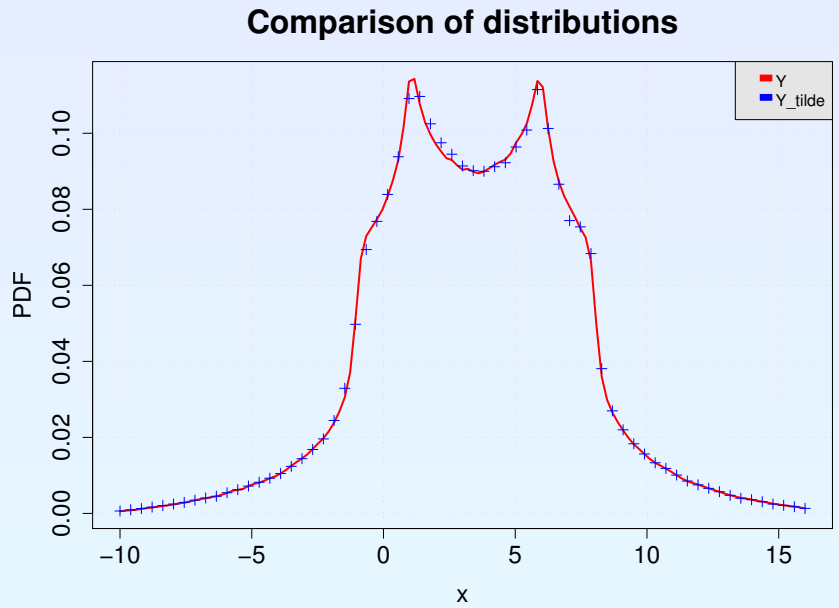
## Moyenne, variance, indices de Sobol

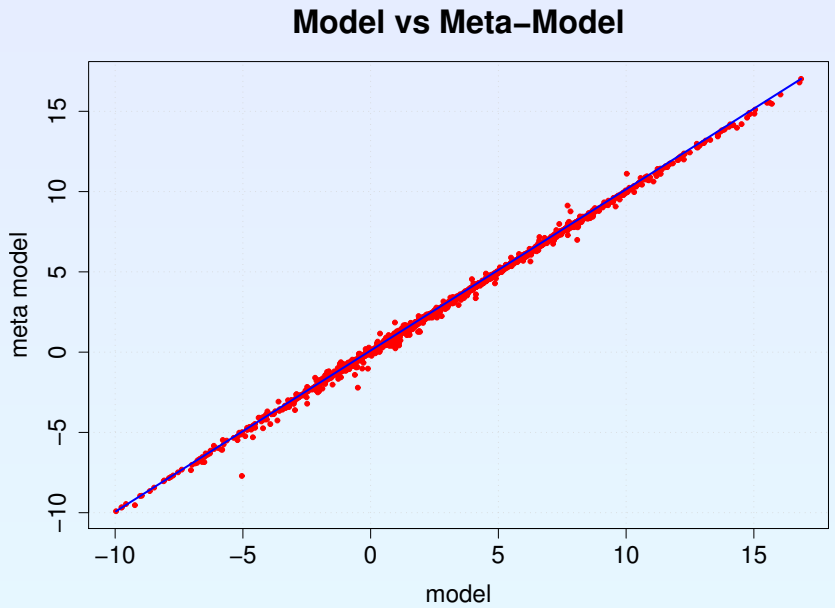
Le budget calcul est limité à 500 évaluations du modèle. La FixedStrategy est donc limitée au degré total 9, car il faut assurer un nombre de calculs au moins double de la taille de la base pour éviter le «surapprentissage» (règle empirique). On obtient :

| | |
|---|---|
| `mean = 3.4915e+00` | relative error=0.2% |
| `variance = 1.3917e+01` | relative error=0.5% |
| `Sobol index 0 = 3.1179e-01` | relative error=0.7% |
| `Sobol index 1 = 4.4344e-01` | relative error=0.2% |
| `Sobol index 2 = 5.2163e-05` | absolute error=5.2e-05 |

L'approximation de ces grandeurs est très bonne. Le méta-modèle fait intervenir 220 coefficients.

# Distribution de $\tilde{Y}$, comparaison modèle/méta-modèle




Les approximations de la distribution et du modèle sont très bonnes. Quelques points diffèrent sensiblement mais la corrélation linéaire entre le modèle et le méta-modèle vaut 1 à la précision machine.

# Valeurs numériques (CleaningStrategy)

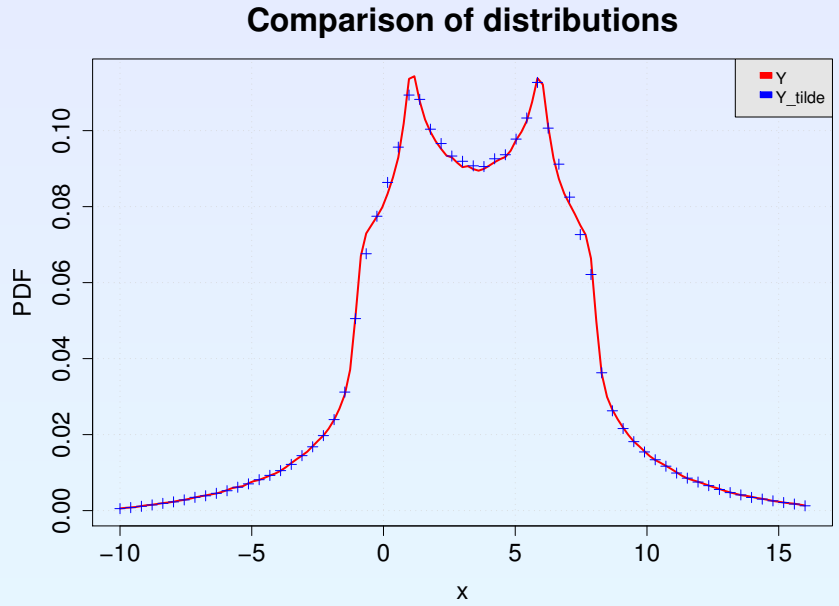
## Moyenne, variance, indices de Sobol

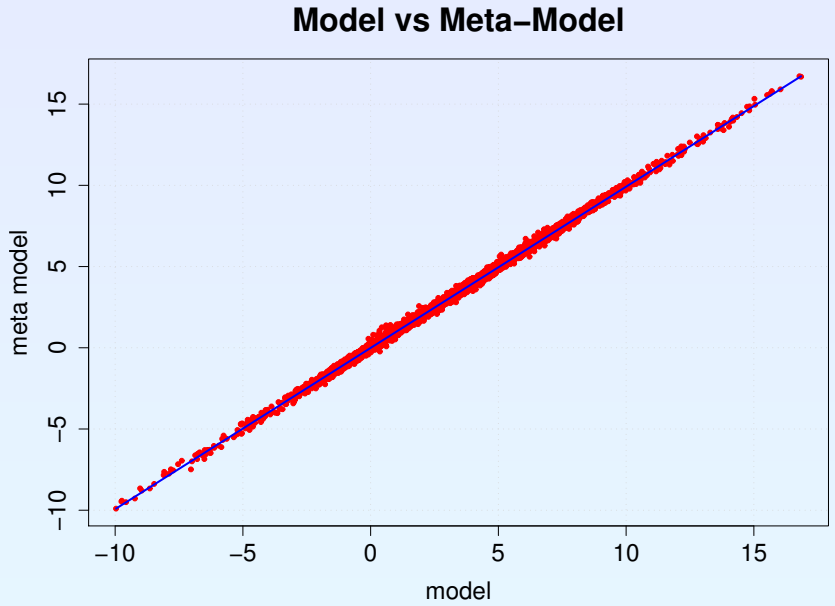
Le budget calcul est limité à 500 évaluations du modèle. La StrategyStrategy n'a pas les mêmes limites que la FixedStrategy : on conserve les 16 coefficients les plus significatif, en explorant les degrés totaux jusqu'à 12 (contributions négligeables au-delà). On obtient :

| | |
|---|---|
| `mean = 3.4990e+00` | relative error=0.0% |
| `variance = 1.3704e+01` | relative error=1.0% |
| `Sobol index 0 = 3.1387e-01` | relative error=0.0% |
| `Sobol index 1 = 4.4189e-01` | relative error=0.1% |
| `Sobol index 2 = 5.8044e-07` | absolute error=5.8e-07 |

L'approximation de ces grandeurs est très bonne. Le méta-modèle fait intervenir 16 coefficients.

# Distribution de $\tilde{Y}$, comparaison modèle/méta-modèle




Les approximations de la distribution et du modèle sont très bonnes. Le nuage est légèrement plus épais que dans le cas de la FixedStrategy, par contre il n'y a plus de points qui s'éloignent de la diagonale. La corrélation linéaire entre le modèle et le méta-modèle vaut 1 à la précision machine.

# Conclusion

## Le chaos polynomial : un outil puissant disponible dans Open TURNS

- Une mise en œuvre qui mime les mathématiques sous-jacentes
- De nombreuses options disponibles
- Une implémentation validée
- Communication scientifique à la conférence ICOSSAR 2009

## Lire la documentation !

- Le Reference Guide pour la théorie
- Le User Manual pour les commandes détaillées
- Le UseCase Guide pour être rapidement opérationnel
- L'Examples Guide pour un exemple de mise en œuvre complète