Polynômes de chaos sous Scilab via la librairie NISP

Michaël Baudin, Jean-Marc Martinez

Polynômes de chaos sous Scilab via la librairie NISP

Michael Baudin — *Consortium Scilab, Digitéo* — *michael.baudin@scilab.org*
Jean-Marc Martinez — *CEA DEN/DANS/DM2S* — *jean-marc.martinez@cea.fr*

Résumé : Cette note présente les polynômes de chaos, adaptés à la modélisation et propagation d'incertitudes en simulation numérique. Ce projet prend appui sur la librairie NISP (Non Intrusive Spectral Projection) développée sous licence LGPL. Le module NISP est disponible dans Scilab [1] sous Linux et Windows et peut être téléchargé via le portail ATOMS administré par le Consortium Scilab. Ce travail a été réalisé dans le cadre du projet OPUS [2] (Open-source Platform for Uncertainty treatment in Simulation) soutenu par l'Agence Nationale de la Recherche. Dans ce document, nous introduisons les polynômes de chaos, ainsi que leur intérêt en analyse de sensibilité comme modèles adaptés à la décomposition fonctionnelle de la variance. Puis nous présentons un tutoriel d'introduction au module NISP de Scilab.

Abstract : In this document, we present chaos polynomials in the context of the modelisation and propagation of uncertainty in numerical simulations. This project is based on the NISP (Non Intrusive Spectral Projection) library, which is developed under the LGPL licence and available in Scilab. The NISP module is available in Scilab [1] for Linux and Windows platforms and can be downloaded from the ATOMS portal, managed by the Scilab Consortium. This project has been performed in the context of the OPUS [2] project (Open-source Platform for Uncertainty treatment in Simulation), funded by the Agence Nationale de la Recherche. The first part of the document is devoted to the introduction of chaos polynomials, where we emphasize their use in sensitivity analysis to compute a functionnal decomposition of the variance. In the second part, we give a tutorial introduction to the use of the NISP module in Scilab.

Mots clés : Statistique mathématique, Enseignement de la statistique, Traitement des incertitudes.

1 Chaos polynomial et Analyse de sensibilité

Cette section introduit les polynômes de chaos et fait le lien avec l'analyse de sensibilité basée sur la décomposition de la variance [3].

1.1 Polynôme de chaos

Les polynômes de chaos issus des travaux de Norbert Wiener permettent de modéliser toute variable aléatoire ou processus de variance finie par un développement en série de la forme

$$X(\xi) = \sum_{\alpha} x_\alpha \Psi_\alpha(\xi) \tag{1}$$

où ξ est un vecteur gaussien de variables indépendantes $\xi = (\xi_1, \xi_2, \ldots)$, les x_α sont les coefficients du développement et les Ψ_α des polynômes multidimensionnels obtenus par tensorisation de polynômes orthogonaux de Hermite $H_{\alpha_i}(\xi_i)$ de degré α_i : $\Psi_\alpha(\xi) = \prod_i H_{\alpha_i}(\xi_i)$. Le multi-indice

$\alpha = (\alpha_1, \alpha_2, \ldots)$ est composé des degrés $\alpha_i = 1, 2, \ldots$ avec $\|\alpha\|_1 = \sum_i \alpha_i$ le degré du polynôme Ψ_α. Ces modèles ont été popularisés par Ghanem comme outils de modélisation et de propagation des incertitudes dans le domaine de la simulation numérique. Cette approche dans la modélisation des incertitudes a été généralisée par [4] aux développements en série basés sur une large classe de polynômes orthogonaux (Hermite, Legendre, Laguerre, Jacobi, ...) afin d'assurer une meilleure convergence dans la modélisation de variables ou processus non gaussiens. En notant $\Phi^i_{\alpha_i}(\xi_i)$ le polynôme orthogonal associé à la variable aléatoire ξ_i et de degré α_i, les polynômes Ψ_α du développement en polynômes de chaos généralisé s'expriment par : $\Psi_\alpha(\xi) = \prod_i \Phi^i_{\alpha_i}(\xi_i)$. Cette représentation simplifie certaines analyses d'incertitudes et de sensibilité effectuées sur les variables aléatoires fonction des variables stochastiques ξ_i. Pour le démontrer, nous considérons l'espace des variables aléatoires de variance finie et fonction des variables aléatoires indépendantes $\xi_1, \xi_2, \ldots$. Cet espace, muni du produit scalaire $< X, Y >= E(XY)$, est un espace de Hilbert où les variables aléatoires Ψ_α constituent une base orthogonale. En effet, en posant $\Phi^i_0(\xi_i) = 1$ les polynômes de degré 0 et en rappelant l'indépendance des variables stochastiques ξ_i et l'orthogonalité des $\Phi^i_{\alpha_i}$, on a $< \Psi_\alpha, \Psi_{\alpha'} >= \prod_i < \Phi^i_{\alpha_i}, \Phi^i_{\alpha'_i} >= \prod_i ||\Phi^i_{\alpha_i}||^2 \delta_{\alpha_i,\alpha'_i} = ||\Psi_\alpha||^2 \delta_{\alpha,\alpha'}$. Les coefficients modaux x_α du développement peuvent être obtenus à partir des produits scalaires entre la variable X et les fonctions polynomiales Ψ_α de la base. Par la suite pour simplifier les expressions, on considère une normalisation des polynômes orthogonaux $\Phi^i_{\alpha_i}$ et donc une normalisation des polynômes Ψ_α. Les coefficients modaux s'obtiennent alors très simplement par $x_\alpha =< X, \Psi_\alpha >$. En notant $\mathbf{0}$ le multi-indice α de composantes nulles et les relations de dominance $\beta \preceq \alpha \Rightarrow \forall i, \beta_i \leq \alpha_i$, et $\beta \prec \alpha \Rightarrow \beta \preceq \alpha, \exists i\ \beta_i < \alpha_i$, on a : $E(X) = x_{\mathbf{0}}, Var(X) = \sum_{\mathbf{0} \prec \alpha} x^2_\alpha$, et la covariance entre deux variables X et Y développées dans la même base par $Cov(X, Y) = \sum_{\mathbf{0} \prec \alpha} x_\alpha y_\alpha$.

La mise en oeuvre des polynômes de chaos nécessite une troncature sur le degré maximal du polynôme. En notant nx la dimension stochastique, c'est à dire le nombre de variables ξ_i et no le degré maximal, le nombre de coefficients du développement est C^{no}_{nx+no} et le développement associé s'écrit $X(\xi) = \sum_{\|\alpha\|_1 \leq no} x_\alpha \Psi_\alpha(\xi)$. Cette complexité combinatoire explique la limitation de l'utilisation des polynômes de chaos aux problèmes à faible dimension. Pour pallier cette limitation, des méthodes ont été proposées afin de réduire le nombre de termes de la série. Dans [5] l'approche consiste à sélectionner les polynômes de faible ordre d'interaction par la contrainte suivante sur le multi-indice $(\sum_{i=1}^{nx} \alpha_i^q)^{q^{-1}} \leq no$. Cette méthode nécessite le choix a priori de la constante $q \in]0, 1]$, le nombre de fonctions polynomiales est d'autant plus petit que q est petit. Dans [6] la méthode consiste à développer les dimensions stochastiques de façon non isotrope où les polynômes orthogonaux $\Phi^i_{\alpha_i}$ n'ont pas nécessairement le même degré maximal. L'intérêt de cette méthode est d'être basée sur une approche adaptative de l'intégration multi-dimensionnelle utilisée pour calculer les coefficients du développement.

Les coefficients peuvent être calculés par intégration numérique ou approximation par moindres carrés. En notant $\{(\xi^j, \omega^j), j = 1, 2, \ldots, n\}$ les points et poids de la formule d'intégration, on a : $x_\alpha =< X, \Psi_\alpha >= E(X, \Psi_\alpha) \simeq \sum_{j=1}^n \omega^j X(\xi^j) \Psi_\alpha(\xi^j)$. Pour une intégration de type Monte Carlo, les points ξ^j sont obtenus à partir de n tirages indépendants du vecteur ξ avec $\omega^j = n^{-1}$. Pour une intégration de type Gauss, les points et poids sont obtenus par tensorisation complète de nx formules de quadrature mono-dimensionnelle d'un niveau suffisant pour intégrer les polynômes de degré no. À grande dimension, le nombre de points de la tensorisation

complète devient rédhibitoire pour la simulation numérique. Une solution est apportée par les constructions de Smolyak [6]. Ces méthodes sont basées sur les différences entre quadratures mono-dimensionnelles de niveaux successifs. Lorsque les formules sont emboîtées, c'est à dire lorsque les points d'une quadrature de niveau $l-1$ sont contenus dans celle de niveau l, les formules de Smolyak sont très efficaces. Pour $nx = 8$ variables aléatoires et un calcul exact des coefficients si la fonction est un polynôme de degré 5, la quadrature tensorisée nécessite $(5+1)^8 = 1679616$ simulations alors que la cubature proposée par Petras (2003) n'exige que 6657 simulations (l'exactitude n'est obtenue que pour des densités uniformes).
Pour les méthodes par moindres carrés, on note les vecteurs $\mathbf{x}, X$ de composantes respectives $x_\alpha, X(\xi^j)$ et Z la matrice d'éléments $Z_{j,\alpha} = \Psi_\alpha(\xi^j)$. La solution, éventuellement régularisée, est obtenue à partir de $\mathbf{x} = (Z^T Z + \lambda I)^{-1} Z^T X$ où le paramètre de régularisation λ peut être instantié par validation croisée. La méthode *Least Angle Regression* proposée par [7] a été développée et adaptée par [5] à la sélection des polynômes Ψ_α les plus corrélés aux réponses. Pour cette classe de méthodes, un des problèmes posés est le choix de l'échantillon à réaliser. Des techniques de planification robuste peuvent être utilisées de façon à minimiser l'erreur d'approximation [8] ou de planification adaptative basée sur l'apprentissage actif [9].

1.2 Analyse de sensibilité

Dans cette partie, on présente l'analyse de sensibilité globale dont l'objectif est de hiérarchiser les paramètres les plus influents sur l'incertitude des résultats. Cette analyse est basée sur la décomposition de la variance. On montre qu'elle s'exprime à partir des coefficients du polynôme. Considérons le modèle :

$$Y = f(X_1, X_2, \ldots, X_d), \tag{2}$$

où d est la dimension du problème et X_i des variables aléatoires de lois connues. On suppose que f est une fonction déterministe et que les variables Y et X_i sont de moyennes et de variances finies. L'outil adapté à l'analyse de la variance de Y est basé sur la décomposition fonctionnelle due à Hoeffding. Soit $U = \{0,1\}^d$ l'ensemble des multi-indices de dimension d. Soit $u \in U$ un multi-indice et f_u une fonction qui ne dépend que des composantes de X dont l'indice est contenu dans u. Si l'on note $f_{\mathbf{0}}$ la fonction constante, la décomposition fonctionnelle est :

$$f(X_1, X_2, \ldots, X_d) = \sum_{u \in U} f_u(X_u). \tag{3}$$

Cette décomposition est unique si, pour tout $X_i \in X_u$, on a $E(f_u|X_i) = 0$. On alors $E(f_u f_v) =< f_u, f_v >= \delta_{u,v}||f_u||^2$. La décomposition de la variance de Y s'obtient alors par : $Var(Y) = \sum_{\mathbf{0} \prec u} \sigma_u^2$ où $\sigma_u^2 = ||f_u||^2$ représente la part de la variance des variables de X_u agissant en interaction. La part totale de la variance de Y due à X_u et l'indice de Sobol S_u associé sont :

$$V[E(Y|X_u)] = \sum_{\mathbf{0} \prec v \preceq u} \sigma_v^2 \quad , \quad S_u = \frac{V[E(Y|X_u)]}{V(Y)} = \frac{\sum_{\mathbf{0} \prec v \preceq u} \sigma_v^2}{\sum_{\mathbf{0} \prec v} \sigma_v^2}. \tag{4}$$

Cette décomposition appliquée à un développement en chaos polynomial est immédiate. En effet grâce à l'orthogonalité des polynômes, les différentes variances σ_u^2 s'obtiennent explicitement à

partir des coefficients du développement. Si le modèle (2) est un polynôme de chaos on a :

$$f_u(X_u(\xi_u)) = \sum_{\min(\alpha,\mathbf{1})=u} y_\alpha \Psi_\alpha(\xi_u) \Rightarrow \sigma_u^2 = \sum_{\min(\alpha,\mathbf{1})=u} y_\alpha^2 \text{ et } S_u = \frac{\sum_{\mathbf{0}\prec\min(\alpha,\mathbf{1})\preceq u} y_\alpha^2}{\sum_{\mathbf{0}\prec\alpha} y_\alpha^2}, \tag{5}$$

et on obtient donc la décomposition fonctionnelle de la variance de Y et les indices de sensibilité.

2 Scilab - Nisp

Le module NISP est compatible avec la version 5.2 de Scilab. Scilab dispose d'un système de modules qui permet d'étendre ses fonctionnalités. Depuis la version 5.2 de Scilab, le système ATOMS permet de chercher, télécharger et installer automatiquement des modules mis à disposition par les utilisateurs de Scilab sur le portail ATOMS, administré par le consortium Scilab. Ainsi, le module Scilab-NISP est disponible à l'adresse suivante http://atoms.scilab.org/toolboxes/NISP/2.1. Le module NISP est disponible sous forme binaire pour Linux 32 et 64 bits ainsi que pour Windows 32 bits. Pour installer le module avec ATOMS, il est nécessaire d'avoir une connection internet. Automatiquement, le système va récupérer la version binaire correspondant au système d'exploitation de la machine. Dans la session Scilab suivante, nous utilisons la fonction *atomsInstall()* pour télécharger et installer le module puis, nous chargeons le module avec la fonction *atomsLoad()*.

```
-->atomsInstall ( "NISP" );
-->atomsLoad("NISP");
```

Le module est immédiatement opérationnel et sera automatiquement chargé au démarrage de la prochaine session. Une documentation en ligne fournit des exemples illustrant les fonctionnalités de Nisp. On pourra consulter les documents présentés lors du 3ème WorkShop Opus (25/11/2009) dédié aux polynômes de chaos sur le site [2] pour plus de détails.

2.1 Cas test de Ishigami

Dans cette section, nous utilisons le module Scilab pour réaliser l'analyse de sensibilité du cas test de Ishigami défini par le modèle suivant :

$$Y = f(X_1, X_2, X_3) = \sin(X_1) + 7\sin(X_2)^2 + 0.1\sin(X_1)X_3^4, \tag{6}$$

où (X_1, X_2, X_3) sont des variables aléatoires uniformes dans $[-\pi, \pi]$. Nous allons approcher le modèle par un développement en chaos polynomial issu de la tensorisation de 3 polynômes de Legendre. La fonction *Ishigami* suivante implémente la fonction f en Scilab.

```
function y = ishigami (x)
  a=7.; b=0.1; s1=sin(x(1)); s2=sin(x(2));
  y(1) = s1 + a*s2^2 + b*s1*x(3)^4;
endfunction
```

La classe *randvar* permet de gérer des variables aléatoires, spécifiées par leur loi de distribution et leurs paramètres. La classe *setrandvar* permet de gérer une collection de variables aléatoires. Nous commençons par définir la collection de variables stochastiques *srvx*, formée de trois variables uniformes dans l'intervalle $[0, 1]$.

```
srvx = setrandvar_new ( 3 );
```

Puis, nous définissons les paramètres *rvu1*, *rvu2* et *rvu3*, uniformes dans l'intervalle $[-\pi, \pi]$.

```
rvu1 = randvar_new ( "Uniforme" , -%pi , %pi );
rvu2 = randvar_new ( "Uniforme" , -%pi , %pi );
rvu3 = randvar_new ( "Uniforme" , -%pi , %pi );
```

Dans le script suivant, nous définissons la collection de variables aléatoires incertaines *srvu*, puis, nous ajoutons les variables aléatoires incertaines dans la collection.

```
srvu = setrandvar_new();
setrandvar_addrandvar ( srvu , rvu1 );
setrandvar_addrandvar ( srvu , rvu2 );
setrandvar_addrandvar ( srvu , rvu3 );
```

Sur la base de la collection de variables stochastiques *srvx*, nous construisons par le script suivant un plan d'expériences. Ce plan est construit sur la base de la méthode d'intégration utilisant la librairie de Petras [10], d'où le nom de l'option.

```
degre = 9;  setrandvar_buildsample ( srvx , "Petras" , degre );
```

Puis, par des transformations probabilistes, le plan d'expériences construit sur les variables stochastiques est transformé en plan d'expériences sur les paramètres incertains.

```
setrandvar_buildsample ( srvu , srvx );
```

La classe "polychaos" permet de gérer un polynôme de chaos. Dans le script suivant, on crée le polynôme *pc* en lui transmettant la collection de variables stochastiques *srvx* (ce qui définit la base du développement) et le nombre de paramètres de sortie.

```
pc = polychaos_new ( srvx , 1 );
polychaos_setdegree ( pc , degre);
```

Nous pouvons désormais connaître le nombre d'expériences à réaliser et nous le stockons dans la variable *np*. Ici, *np*=751, valeur transmise au polynôme de chaos, dans le script suivant.

```
np = setrandvar_getsize ( srvu );
polychaos_setsizetarget ( pc , np );
```

Dans le script suivant nous réalisons les simulations spécifiées par *srvu*.

```
for k=1:np
  inputdata = setrandvar_getsample ( srvu , k );
  outputdata = ishigami ( inputdata );
  polychaos_settarget ( pc , k , outputdata );
end
```

Puis, nous calculons les coefficients du polynôme de chaos par intégration.

```
polychaos_computeexp ( pc , srvx , "Integration" );
```

Nous pouvons maintenant éditer moyenne et variance.

```
-->average = polychaos_getmean(pc)
 average  = 3.5
-->var = polychaos_getvariance(pc)
 var  = 13.842473
```

La session suivante permet d'afficher les indices de sensibilité du premier ordre ainsi que les indices de sensibilité totaux.

```
-->disp('Indice du 1er ordre :'); disp(polychaos_getindexfirst(pc)')
 Indice du 1er ordre : 0.3139532    0.4423253    8.086D-31
-->disp('Indice Totaux :'); disp(polychaos_getindextotal(pc)')
 Indice Totaux : 0.5576747    0.4423255    0.2437215
```

3 Conclusions

L'objectif de cette étude visait certains objectifs du projet OPUS, sur le développement de modules spécifiques (les contributions), sur la mise à disposition des utilisateurs de Scilab de ces modules et sur le volet de l'enseignement. En effet le langage Scilab nous semble suffisamment intuitif pour que l'enseignement puisse se concentrer davantage sur la méthode que sur les aspects informatiques. Les développements futurs du module NISP porteront sur l'introduction d'autres classes de polynômes et lois de probabilité.

References

[1] Consortium Scilab. Plateforme open source de calcul scientifique. http://www.scilab.org.

[2] OPUS Consortium. Opus project. http://opus-project.fr.

[3] Th. Crestaux, O. Le Maître, and J.M. Martinez. *Polynomial chaos expansion for sensitivity analysis. Reliability Engineering and System Safety*, 94:1161–1172, 2009.

[4] D. Xiu and G. E. Karniadakis. Modelling uncertainty in flow simulations via generalized polynomial chaos. *Journal of Computational Physics*, 187:137–167, 2003.

[5] Blatman G. Adaptative sparse polynomial chaos expansion for uncertainty propagation and sensitivity analysis. *PhD University Blaise-pascal, Clermont II*, 2009.

[6] Th. Gerstner and M. Griebel. Dimension-adaptive tensor-product quadrature. *Computing*, 71(1):65–87, 2000.

[7] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[8] F. J. Yue, R. X. et Hickernell. Robust designs for fitting linear models with misspecification. *Statist. Sinica 9*, 9:1053–1069, 1999.

[9] S. Gazut, Martinez J.M., G. Dreyfus, and Y. Oussar. Towards the optimal design of numerical experiment. *IEEE Trans. on Neural Networks*, 19(5):874–882, 2008.

[10] K. Petras. On the smolyak cubature error for analytic functions. *Advances in Computational Mathematics*, 12:71–93, 2000.