OpenTURNS, logiciel Open Source pour le traitement des incertitudes dans un contexte industriel

OpenTURNS, an Open Source software for the treatment of uncertainties in an industrial context

Michaël Baudin, Anne Dutfoy
EDF R&D
Management des Risques Industriels
6, quai Watier
78401 Chatou Cedex

Résumé
Nous présentons le logiciel open source OpenTURNS, qui permet la modélisation et le traitement des incertitudes paramétriques entachant les données d'entrée d'un code déterministe.
Summary
We analyze the open source software OpenTURNS, which models and propagates the uncertainties of the inputs through a deterministic function.

1 Contexte

EDF et son Groupe, en tant qu'opérateur de grandes installations de production d'énergie est soumis à la fois à des exigences particulières de sûreté et de maîtrise environnementale, et aux aléas des marchés (demande et offre concurrente) et des ressources (liées au climat, à l'environnement, ...). Pour satisfaire à ces exigences et développer sa performance, l'entreprise doit travailler sur ses marges, en vue de les réduire, tout en contrôlant sa prise de risques et en optimisant le cycle de vie de ses installations. Actuellement, il devient possible de fournir des prédictions quantitatives crédibles susceptibles d'améliorer significativement les processus de décision et la création de valeur. La prise en compte des incertitudes permet d'accroître la qualité des prédictions et participe donc très fortement aux enjeux majeurs du groupe EDF dans toute leur diversité. Les méthodes probabilistes pour le traitement des incertitudes visent à faire gagner des marges tout en confortant le respect des enjeux du Groupe.
A l'instar d'EDF, les industriels prennent en compte les incertitudes dans leurs études, afin d'adosser leur décision à une prise de risque quantifiée. Depuis quelques années, plusieurs d'entre eux partagent leurs pratiques autour d'une méthodologie générique, commune à tous les métiers, qui permet la capitalisation des connaissances et facilite l'acceptation par des autorités de contrôle et de certification de cette nouvelle approche probabiliste.

2 Méthode

2.1 Introduction

C'est dans ce contexte que les trois industriels, EDF R&D, EADS IW et Phimeca Engineering, ont construit un partenariat pour la conception d'un outil open source de traitement des incertitudes en simulation numérique, OpenTURNS [EADS et al., 2014b], reposant sur la théorie des probabilités. Ce projet, initié en 2005, a permis d'élaborer une plate-forme logicielle répondant aux exigences de :
- transparence : OpenTURNS, afin d'être accepté et utilisé par une large communauté d'ingénieurs et experts, vise un affichage explicite des méthodologies mises en oeuvre, via une diffusion open source ;
- performance : OpenTURNS est conçu pour exploiter au maximum les capacités de calcul disponibles afin d'utiliser au mieux les ressources multi-coeurs et HPC, particulièrement pertinentes dans le cadre des études probabilistes où un grand nombre de calculs est lancé ;
- généricité : OpenTURNS se veut la déclinaison logicielle de la méthodologie de traitement d'incertitudes partagée par un grand nombre d'industriels, commune à tous les métiers.

La plate-forme OpenTURNS permet d'utiliser une méthodologie de traitement des incertitudes dans l'ensemble de ses étapes [Pasanisi and Dutfoy, 2012] :
- Etape A : Spécification de l'étude
- Etape B : Quantification des sources d'incertitudes : modélisation de la loi du vecteur aléatoires des variables incertaines ;
- Etape C : Propagation d'incertitudes pour quantifier le critère sélectionné en étape A via des méthodes de simulation ou analytiques ;

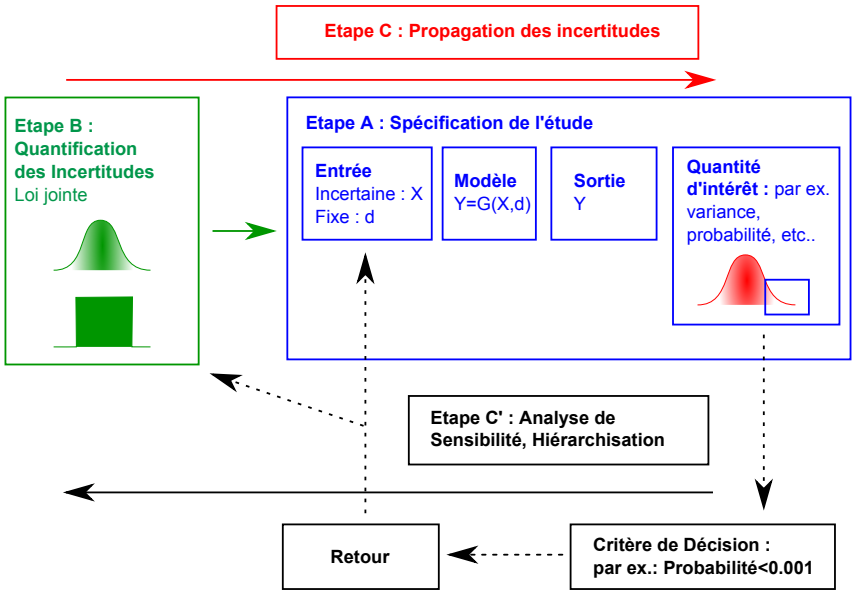

FIGURE 1 – Les différentes étapes de la méthodologie du traitement d'incertitude.

– Etape C' : Hiérarchisation des incertitudes sur la quantification du critère.

La figure 1, issue des travaux du groupe "Incertitudes et Industrie" de l'IMDR, présente les quatre principales étapes de la méthodologie.

2.2 Etape A : spécification de l'étude

Cette étape consiste à définir (ou choisir) les différentes composantes de l'étude. On considère le code de calcul G pour l'évaluation de la variable d'intérêt (éventuellement vectorielle) Y. Les entrées du code considérées avec certitude d sont séparées des entrées incertaines X, modélisées par un vecteur aléatoire. On définit enfin la caractéristique de la distribution de probabilité de Y (dite quantité d'intérêt), dont l'évaluation est attendue en fonction des finalités de l'étude. On peut distinguer plusieurs types de quantités d'intérêt :

– la tendance centrale (moyenne, écart-type), par exemple, dans un problème de métrologie.
– minimum ou maximum, c'est à dire l'étendue de la sortie Y étant donné la variabilité des entrées aléatoires X,
– la probabilité de dépassement de seuil (un quantile), par exemple dans un problème de sûreté.

Cette étape, cruciale, oriente de façon très importante le choix des algorithmes permettant d'estimer les résultats.

2.3 Etape B : quantification des sources d'incertitude

L'objectif de cette étape est de déterminer la loi du vecteur des entrées X. On doit alors déterminer à la fois les lois marginales $f_i(X_i)$ de chaque composante X_i, ainsi que la structure et les paramètres de la dépendance entre les composantes. Cette dépendance peut s'estimer, par exemple, par une matrice de variance-covariance (à l'aide, par exemple, du coefficient de corrélation de Pearson), ou, de manière plus générale, à l'aide de la théorie des copules.

L'identification des lois marginales se fait en fonction de la quantité de données disponibles. Dans le cas où les données sont peu nombreuses, il est nécessaire de faire appel à des experts, dont l'avis est ensuite traduit en termes de loi de distribution. Lorsque les données sont suffisamment nombreuses, on peut utiliser les techniques classiques d'inférence statistique, en utilisant une approche paramétrique ou non paramétrique.

2.4 L'évaluation de G

L'évaluation de la fonction G pose deux difficultés pratiques liées d'une part au pilotage du code par le logiciel de traitement des incertitudes et d'autre part au coût CPU d'évaluation de la fonction G.

En pratique, il est nécessaire d'être capable d'évaluer la fonction G en des points X déterminés par un plan d'expérience fixé à l'avance (dans le cas, par exemple, de la méthode Monte-Carlo simple), ou séquentiel (dans le cas, par exemple, d'une optimisation numérique). C'est pourquoi la pratique nécessite de connecter le logiciel de traitement des incertitudes au code de calcul. Lorsque le code G est un exécutable, ce jumelage peut se faire via des fichiers textes ou binaires. Par ailleurs, lorsque le code G est une librairie, via l'interface de programmation du code par des échanges en mémoire. Cette étape sera revue plus en détail dans la suite.
D'autre part, la méthodologie peut s'appliquer aussi bien à une fonction G analytique de faible coût d'évaluation (par exemple, inférieur à une seconde) qu'à un code G plus complexe, de coût d'évaluation élevé (par exemple, de quelques minutes à quelques heures). Ainsi, l'évaluation de G peut impliquer le calcul d'un système d'équations non linéaires ou encore l'état stationnaire ou transitoire d'un système d'équations différentielles ou aux dérivées partielles mono ou multidimensionnel.
Évidemment, ces différentes situations ne mènent pas au même choix des algorithmes de résolution lors de l'étape B de quantification. Lorsque le coût d'évaluation est faible (par exemple, inférieur à la seconde), il est suffisant d'utiliser les processeurs multi-coeurs qui équipent l'immense majorité des ordinateurs d'aujourd'hui. Lorsque le coût d'évaluation est élevé (par exemple, supérieur à une heure), on peut avoir recourt à des calculateurs haute performance utilisant des supercalculateurs distribués.

2.5 Etape C : propagation des incertitudes

L'étape de propagation des incertitudes consiste à transférer l'incertitude de l'entrée X vers la sortie Y via le code numérique G, et à estimer la quantité d'intérêt. Cette étape mène à la sélection d'un algorithme de résolution, dont le choix est guidé, entre autres critères, par l'objectif visé et par le coût d'évaluation de la fonction G.
En ce qui concerne une étude en tendance centrale, si la fonction G est d'évaluation peut coûteuse, on peut utiliser un échantillonage de type Monte-Carlo simple, ce qui fournit également des intervalles de confiance pour les résultats. Pour une analyse en tendance centrale, la méthode du cumul quadratique (fondée sur un développement de Taylor) est une alternative économique qui peut s'avérer utile lorsque la fonction G est coûteuse à évaluer. Toutetois, cette méthode ne permet pas d'obtenir des intervalles de confiance. D'autres méthodes de Monte-Carlo accélérées sont disponibles pour évaluer la tendance centrale, comme par exemple les plans d'expériences de type Latin Hypercube Sampling (LHS) ou encore les séquences à faible discrépance (Quasi Monte-Carlo).
Pour une étude de type minimum-maximum, on peut recourir à des algorithmes d'optimisation numérique ou à des techniques d'échantillonnage aléatoire ou déterministes.
Pour évaluer une probabilité de dépassement de seuil, il est classique d'utiliser des techniques de simulation comme, par exemple, la méthode de Monte-Carlo. D'autres techniques sont disponibles comme le tirage d'importance ou l'échantillonage directionnel classique ou adaptatif [Munoz-Zuniga, 2011]. Ces dernières techniques fournissent, de plus, un intervalle de confiance sur l'estimation de la probabilité. Fondée sur un algorithme de Metropolis-Hastings modifié, la méthode par Subset Simulation [Au and Beck, 2001] est une autre méthode d'évaluation de probabilité faible. Lorsque la fonction G est très coûteuse, on peut avoir recourt à la méthode FORM-SORM [H.O. Madsen, 1986] fondée sur un développement de Taylor et issue du domaine de la fiabilité des structures.

2.6 Etape C' : hiérarchisation des incertitudes

L'étape de hiérarchisation des incertitudes consiste à ordonner les variables d'entrée X_i par ordre décroissant d'importance en fonction d'un critère et en fonction des finalités de l'étude.
Par exemple, en tendance centrale, les indices basés sur les coefficients de régression d'un modèle linéaire (Standardized Regression Coefficients) peuvent être utilisés pour hiérarchiser la contribution de chaque variable d'entrée à la variance de Y. Toujours en tendance centrale, les indices de sensibilité de Sobol permettent d'étendre cette approche dans le cas général d'un modèle non linéaire.
Lorsqu'on s'intéresse au dépassement d'un seuil, la méthode FORM-SORM permet d'évaluer des facteurs d'importance qui permettent de quantifier l'impact d'un paramètre sur la probabilité de dépassement.

2.7 Utilisation d'un méta-modèle

Il peut être intéressant d'approcher la fonction G par un modèle de remplacement, souvent appelé méta-modèle. Lorsque la fonction G est très coûteuse en temps CPU, le méta-modèle $\tilde{G}$ peut être intéressant si son évaluation est peu coûteuse. La méthode procède alors en deux étapes. La phase d'apprentissage

consiste, dans un premier temps, à calculer les paramètres du méta-modèle. La phase d'exploitation consiste à utiliser le méta-modèle $\tilde{G}$ en remplacement du modèle G dans l'étape C de propagation des incertitudes.
Les méta-modèles les plus utilisés actuellement dans le domaine du traitement des incertitudes sont les réseaux de neurones, les polynômes de chaos et le krigeage.
De plus, le méta-modèle $\tilde{G}$ peut révéler des informations sur le modèle G, que l'on peut exploiter, par exemple, dans le contexte de l'étape C' de hiérarchisation des sources d'incertitudes. Par exemple, les coefficients du polynôme de chaos permettent d'estimer les indices de sensibilité de Sobol de G.
Le remplacement d'un modèle par son méta-modèle peut se faire en tenant compte de l'objectif de l'étude, en adaptant la stratégie de construction (par exemple, par une technique séquentielle) pour minimiser l'erreur d'estimation de la quantité d'intérêt.

2.8 Mise en oeuvre

L'étape A de spécification de l'étude et de ses finalités est sans aucun doute la plus importante. Ce travail, à réaliser en collaboration entre les experts du métier concerné par l'étude et les porteurs de la compétence plus générique Proba/Stat, permet non seulement de faire en sorte que l'étude réponde à la problématique métier, mais oriente le choix des méthodes et des algorithmes à utiliser, très dépendants du coût CPU d'un appel au code de calcul et de la nature de la quantité d'intérêt (moyenne et écart type ou quantile et probabilité de défaillance).
Le coût CPU d'une étude dépend essentiellement du nombre d'appels au code à réaliser pour avoir une bonne estimation de la quantité d'intérêt, le surcoût induit par les traitements statistiques et probabilistes sur les données d'entrée (à l'exception du cas de l'inversion probabiliste) et de sortie du code étant pratiquement nul, ou du moins négligeable. En pratique un compromis est toujours à établir entre la qualité de l'estimation et le nombre de simulations pouvant être réalisées.

3 Le logiciel OpenTURNS

3.1 Introduction

OpenTURNS est un outil de traitement des incertitudes développé depuis 2005 en partenariat par EDF R&D, EADS Innovation Works et Phimeca Engineering. C'est un outil servant au traitement des incertitudes et, a fortiori, à des études de type « analyse de fiabilité des structures ». Il est adapté à la méthodologie de traitement des incertitudes utilisée à EDF. Il regroupe :
– une bibliothèque C++ avec des méthodes et algorithmes spécifiques au traitement des incertitudes et
– un module Python.
Il permet de procéder à l'analyse de la dispersion centrale, de la probabilité de dépassement d'un seuil, des quantiles, de sensibilité. Dans le cas de modèles physiques complexes (non-analytiques par exemple), il est possible de coupler OpenTURNS avec un code externe, par exemple un code de calcul aux éléments finis. Deux versions par an (en moyenne) d'OpenTURNS sont diffusées par le partenariat. Dans le but d'assurer la maintenance en conditions opérationnelles de l'outil pour les besoins de l'intégration d'OpenTURNS dans la plateforme Salomé, deux versions de maintenances sont réalisées par EDF (avec une cadence semestrielle), dont la diffusion est interne à EDF.
Le logiciel OpenTURNS est distribué sous la licence LGPL pour son code informatique et la licence FDL pour sa documentation.

3.2 Fonctionnalités

Pour la quantification des sources d'incertitudes (étape B), OpenTURNS met à disposition des fonctionnalités de traitement statistique à partir d'échantillons, en particulier des tests d'ajustement paramétriques et non paramétriques. Il permet aussi de modéliser des lois de type combinaison linéaire de densités ou de variables aléatoires. Les dépendances aléatoires sont modélisées à l'aide de la théorie des copules. Le mécanisme de construction de loi multivariée basé sur l'association des marginales et de la copule permet des manipulations complexes des lois de probabilités multivariées (par exemple, l'extraction de copule d'une loi multi-variée reconstruite par noyaux ou encore la composition de copules). OpenTURNS fournit également les processus stochastiques (ARMA, processus gaussiens stationnaires, processus non stationnaires, champs aléatoires), ainsi que la modélisation bayésienne.
Pour la propagation des incertitudes (étapes C), OpenTURNS propose des méthodes de calcul analytiques dès que possible pour la détermination de la loi de probabilité de la variable d'intérêt, basées sur

la manipulation des fonctions caractéristiques ainsi que plusieurs méthodes de simulation et les méthodes FORM et SORM généralisées aux copules elliptiques. OpenTURNS dispose des plans d'expériences déterministes ou probabilistes : Monte-Carlo, Latin Hypercube Sampling, quasi Monte-Carlo, etc... OpenTURNS dispose également de méthodes de simulation avec réduction de variance, comme, par exemple, la simulation directionnelle ou l'échantillonnage d'importance. Depuis 2012, puis 2013, OpenTURNS dispose également de modules externes fournissant des fonctionnalités pour l'estimation de probabilité de dépassement de seuil par la méthode de la stratification directionnelle adaptative (issu de la thèse de Miguel Munoz-Zuniga[Munoz-Zuniga, 2011]), ou par Subset Simulation [Au and Beck, 2001].
Pour la hiérarchisation (étapes C'), OpenTURNS évalue des facteurs d'importance et de sensibilité adaptés aux méthodes de propagation sélectionnées, dont les indices de Sobol.
OpenTURNS offre la possibilité de construire des surfaces de réponse dont le chaos polynomial plein et creux, issus de la thèse de G. Blatman [Blatman, 2009] et le krigeage (dans la version 1.3, décembre 2013). Des modules externes ont étés développés pour étendre les fonctionnalités et fournir d'autres métamodèles (Support Vector Machine, Mixmod pour la classification) ou d'autres algorithmes d'estimation de probabilité de dépassement de seuil (Subset Sampling et Simulation Directionnelle Adaptative). Certains de ces modules externes sont issus de travaux de thèses menées dans le département MRI d'EDF R&D.

3.3 Evaluer G de manière performante

Sur le plan pratique, le logiciel OpenTURNS fournit des fonctionnalités qui permettent de se connecter au code de calcul, puis d'évaluer G de manière performante.
Dans le contexte d'OpenTURNS, la technique qui permet de se connecter à la fonction G est appelée "wrapping" (en français, emballage).
Dans le cas le plus simple, la fonction G est une fonction analytique, fournie sous la forme d'une chaîne de caractère. Dans ce cas, OpenTURNS est capable d'évaluer la fonction G de manière extrêment performante. De plus, dans la plupart des cas, OpenTURNS est capable de calculer automatiquement les dérivées (gradient et matrice Hessienne) de manière exacte, par dérivation symbolique de G. Cette capacité peut s'avérer déterminante dans le cas où l'algorithme inclut une étape d'optimisation (par exemple, FORM-SORM) ou une approximation de G (par exemple, un développement de Taylor dans la méthode du cumul quadratique).
Le cas le plus commun est celui où la fonction G est développée par l'utilisateur sous la forme d'une fonction Python. Le travail est facile (pour celui qui est familier de Python), et permet de tirer parti de la richesse et la puissance de ce langage de programmation, puisqu'on peut utiliser toutes les librairies Python existantes. Par exemple, si le code de calcul utilise des fichiers XML en entrée ou en sortie, il est facile d'utiliser les fonctionnalités XML de Python (par exemple, la librairie `minidom`). Si, comme beaucoup de codes de calcul que nous connaissons, l'échange de données se fait par des fichiers texte (ASCII), OpenTURNS fournit un composant (`coupling_tools`) capable de lire et écrire des fichiers textes structurés par les numéros de lignes et contenant éventuellement des tables de données (structurées par des numéros de colonnes). De plus, OpenTURNS fournit un composant capable d'exécuter une telle fonction Python en utilisant les capacités de traitement multi-coeur (multithread) de l'immense majorité des ordinateurs d'aujourd'hui.
Si le code de calcul G est fourni sous la forme d'une librairie C ou Fortran, OpenTURNS fournit un dispositif de couplage générique permettant d'échanger les données via la mémoire de manière performante et multithread. Ce composant peut être configuré par l'interface Python via un fichier XML. Si la solution précédente n'est pas assez souple, il est possible de réaliser le couplage via une interface en C++.

3.4 Le calcul haute performance

Pour l'ingéniérie du quotidien, OpenTURNS permet d'évaluer G en utilisant les capacités de traitement multi-coeur d'un ordinateur portable ou d'une station de travail scientifique. Pourtant, lorsque le code de calcul est très coûteux et que le nombre d'appels devient important, ces capacités ne sont pas suffisantes et il est nécessaire d'utiliser un supercalculateur haute performance.
Dans ce contexte, deux solutions sont communément utilisées. La première consiste à utiliser une fonctionnalité de l'outil permettant d'exécuter une fonction Python sur des processeurs distants, connectés sur le réseau par `ssh`. Les données sont alors échangées via des fichiers, localisés dans des répertoires séparés générés automatiquement, ce qui rend impossible l'écrasement des données intermédiaires. Cette fonctionnalité (la `DistributedPythonFunction`) permet, sur chaque processeur, d'utiliser ses capacités multi-coeurs.

La seconde consiste à utiliser le composant OpenTURNS intégré à la plateforme Salome. Ce composant permet d'utiliser, en plus des fonctionnalités d'OpenTURNS et d'une interface graphique (appelée Eficas), un composant, appelé YACS, qui permet d'appeler un script Python. Le composant YACS est le module de supervision des calculs de Salome, et fournit à la fois une interface graphique permettant d'enchaîner les calculs en reliant les entrées et les sorties, mais aussi d'exécuter ces calculs sur des machines distantes. Plusieurs études ont été réalisées à EDF sur la base du composant OpenTURNS de Salome. Par exemple, un calcul de propagation d'incertitude (calculs de dimensionnement thermique du stockage des déchets de haute-activité et à vie longue) impliquait un calcul unitaire d'environ 10 minutes sur les 8 processeurs d'une station de travail (en mémoire partagée). Dans le contexte de Salome, le calcul OpenTURNS impliquant 6000 calculs unitaires a nécessité environ 8000 heures CPU, sur 32 noeuds [Barate, 2013].

3.5 Environnement technologique

Développé sous GNU/Linux et porté sur Windows, OpenTURNS peut s'utiliser de trois manières différentes : sous forme de librairie C++ incluse dans une application métier, sous forme de module Python ou bien à l'aide d'une interface graphique basée sur le logiciel open source EFICAS développé par EDF R&D.
OpenTURNS est compatible avec Salomé, plate-forme de liaison CAO-calcul-visualisation développée au sein d'un partenariat dont EDF R&D et CEA sont les principaux développeurs.
La licence du logiciel (LGPL) lui permet d'être utilisé librement dans un contexte commercial. C'est grâce à cette licence que le logiciel PhimecaSoft, distribué par Phimeca Engineering, fournit une interface graphique utilisant OpenTURNS comme moteur de calcul. De plus, PhimecaSoft fournit des modules propriétaires complémentaires à ceux d'OpenTURNS.
Le projet logiciel OpenTURNS est de taille relativement importante. La version 1.3 de l'outil fournit plus de 500 classes, avec environ 4000 fichiers pour le code source et 800 pour la documentation. C'est pourquoi une large part du projet est consacrée à la documentation et à la formation, ce qui est le sujet de la section suivante.

3.6 Documentation et formation

La documentation d'OpenTURNS est disponible sur le site du logiciel [EADS et al., 2014b]. Elle est fournie en anglais.
La documentation d'OpenTURNS v1.3, disponible au format HTML ou PDF, est constituée des documents suivants [EADS et al., 2014a] :
- Reference Guide : Ce document présente la méthodologie de traitement des incertitudes.
- Examples Guide : Ce document présente plusieurs exemples d'études dans lesquels la méthodologie est mise en oeuvre.
- Use Cases Guide : Ce document, trié par cas-test, présente des extraits de scripts Python mettant en oeuvre OpenTURNS.
- User Manual TUI : Ce document présente les classes et méthodes disponibles dans les interfaces C++ et Python.
- Developer's Guide : Ce document orienté vers le développeur, présente l'architecture du logiciel, les règles techniques de contribution, de codage, le portage Windows, et les informations nécessaires à la création d'un connecteur (appelé "wrapper") entre OpenTURNS et le code de calcul G.
- Reference Guide of OpenTURNS' classes and library API (HTML uniquement) : Ce document, destiné au développeur, présente l'interface de programmation complète d'OpenTURNS, du point de vue du langage C++.

L'Institut de Transfert des Technologies (ITech) a pour vocation de diffuser les savoir-faire et les innovations issues de la recherche d'EDF R&D vers le groupe EDF. Il constitue un lieu de partage et d'échange des pratiques et des innovations. Une formation EDF est disponible dans le catalogue Itech intitulé "Traitement des incertitudes : utilisation de l'outil OpenTURNS". Cette formation est ouverte à tous (c'est-à-dire EDF et hors EDF) et dure trois jours. Elle est composée d'exposés et de travaux pratiques. Le contenu du stage est le suivant.
- Présentation de l'outil OpenTURNS : Fiche technique de l'outil, Consortium, site web, journée utilisateurs, ...
- Manipulation d'OpenTURNS via ses interfaces graphique EFICAS et textuelle (Python) : déroulement sur un cas d'étude de l'ensemble de la Méthodologie.
- Réalisation de plusieurs couplages informatiques avec OpenTURNS à travers de nombreux TP.

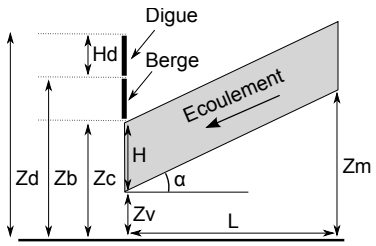

FIGURE 2 – Test Crue - Vue de profil.

Cette formation vient en complément d'une autre formation de trois jours sur les aspects méthodologiques du traitement des incertitudes.

3.7 Exemple d'utilisation

Nous souhaitons estimer le risque de débordement d'une rivière, c'est à dire évaluer un risque d'inondation lors d'une crue. Pour cela, on utilise les équations des écoulements à surface libre issus de l'hydraulique. L'objectif est de calculer une hauteur de surverse S, qui est négative lorsque la rivière est contenue par la digue, et positive lorsqu'il y a une inondation. La figure 2 présente une vue d'ensemble de la situation de profil. Dans cette section, on considère des données fictives, mais réalistes, dont les ordres de grandeur pourraient correspondre à un fleuve français.
On suppose que les paramètres suivants (déterministes) sont donnés :
– hauteur de la digue : $H_d = 3$ (m),
– cote de la berge : $Z_b = 55.5$ (m),
– longueur du tronçon de rivière : $L = 5000$ (m),
– largeur de la rivière : $B = 300$ (m).
On considère les variables aléatoires indépendantes suivantes :
– Q, le débit moyen de la rivière (m^3/s), de loi Gumbel de paramètres 1013 (mode) et 558 (échelle),
– K_s, le coefficient de Manning-Strickler de la rivière $(m^{1/3}/\text{s})$, de loi Normale de paramètres 30 (moyenne) et 7.5 (écart-type),
– Z_v, la cote du fond de la rivière en aval (m), de loi uniforme entre 49 et 51,
– Z_m, la cote du fond de la rivière en amont (m), de loi uniforme entre 54 et 56.
Les variables physiques Q et K_s sont physiquement positives, de telle sorte que les distributions doivent être tronquées.
La pente α (en radians) de la rivière est supposée constante, et faible, de telle sorte que

$$\alpha = \frac{Z_m - Z_v}{L}.$$

Le coefficient de Manning-Strickler de la rivière $(m^{1/3}/\text{s})$, noté K_s, est une caractéristique de la "conductivité" exercée par le fond de la rivière sur l'écoulement. Lorsque le coefficient de Manning-Strickler K_s est plus grand, le fond de la rivière s'oppose moins à l'écoulement. La valeur du coefficient K_s est de 75 à 90 pour du béton lisse, 60 pour un canal en terre, 30 pour une rivière de plaine, 10 à 15 pour un lit majeur urbanisé et inférieur à 10 pour un lit majeur en forêt [Degoutte, 2007].
Le débit de la rivière (m^3/s) est noté Q. Dans un problème de calcul de crue, la variable Q est le débit annuel maximal.
Sous certaines hypothèses, la hauteur d'eau maximale (en mètres) est :

$$H = \left(\frac{Q}{K_s B\sqrt{\alpha}}\right)^{3/5}.$$

La surverse S est la différence entre la hauteur de la crue et la hauteur de la digue, c'est à dire $S = Z_c - Z_d$. Cela implique

$$S = Z_v + H - H_d - Z_b.$$

On considère deux cas en fonction du signe de S : $S < 0$ correspond au domaine de sûreté et $S > 0$ correspond à la défaillance (inondation). On considère le modèle

$$S = G(Q, K_s, Z_v, Z_m).$$

Le script OpenTURNS qui traite ce problème est le suivant. Le script utilise un algorithme de Monte-Carlo simple pour estimer la probabilité d'inondation. On calcule, de plus, un intervalle de confiance.

```
from openturns.viewer import View
from numpy import inf
from openturns import (
    ComposedDistribution, Gumbel,
    RandomVector, Normal, PythonFunction, MonteCarlo,
    TruncatedDistribution, Event, Uniform, GreaterOrEqual
)
from math import sqrt

# 1. The function G
def functionCrue(X) :
    H_d = 3.0;      Z_b = 55.5
    L = 5.0e3;      B = 300.0
    Z_d = Z_b + H_d
    Q, K_s, Z_v, Z_m = X
    alpha = (Z_m - Z_v)/L
    H = (Q/(K_s*B*sqrt(alpha)))**(3.0/5.0)
    Z_c = H + Z_v
    return [Z_c - Z_d]

# Creation of the problem function
f = PythonFunction(4, 1, functionCrue)
f.enableHistory()

# 2. Random vector definition
Q = Gumbel(1./558., 1013.)
Q = TruncatedDistribution(Q, 0, inf)
K_s = Normal(30.0, 7.5)
K_s = TruncatedDistribution(K_s, 0, inf)
Z_v = Uniform(49.0, 51.0)
Z_m = Uniform(54.0, 56.0)

# 3. View the PDF
Q.setDescription(["Q (m3/s)"])
K_s.setDescription(["Ks (m^(1/3)/s)"])
Z_v.setDescription(["Zv (m)"])
Z_m.setDescription(["Zm (m)"])

View(Q.drawPDF()).show()
View(K_s.drawPDF()).show()
View(Z_v.drawPDF()).show()
View(Z_m.drawPDF()).show()

# 4. Create the joint distribution function,
#    the output and the event.
inputRandomVector = ComposedDistribution([Q, K_s, Z_v, Z_m])
outputRandomVector = RandomVector(f, RandomVector(inputRandomVector))
eventF = Event(outputRandomVector, GreaterOrEqual(), 0)

# 5. Create the Monte-Carlo algorithm
algoProb = MonteCarlo(eventF)
algoProb.setMaximumOuterSampling(100000)
algoProb.run()

# 6. Get the results
resultAlgo = algoProb.getResult()
```

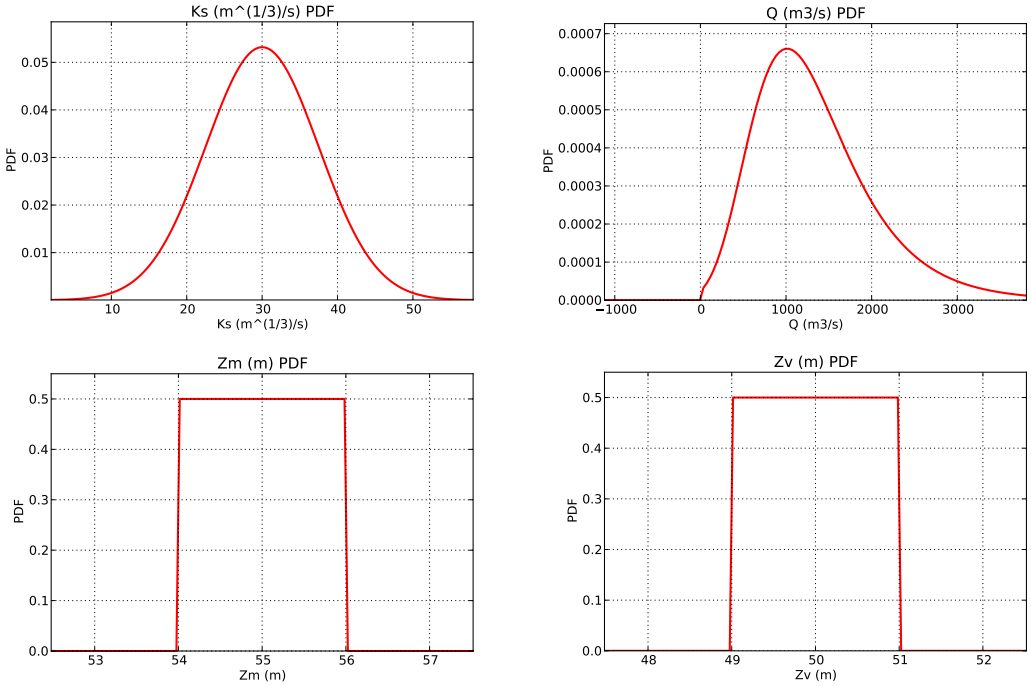

FIGURE 3 – Densité des quatre variables K_s, Q, Z_m et Z_v.

```
Neval = f.getEvaluationCallsNumber()
print "Number of function calls =", Neval
Pf = resultAlgo.getProbabilityEstimate()
print "Failure Probability = %e" % (Pf)
c95 = resultAlgo.getConfidenceLength()
pmin=Pf-0.5*c95
pmax=Pf+0.5*c95
print "within [%e,%e]" % (pmin,pmax)

# 7. Plot the histogram
from openturns import VisualTest
outComputedPoints = f.getOutputHistory().getSample()
histoGraph = VisualTest.DrawHistogram(outComputedPoints,100)
histoGraph.setTitle("Histogramme de la surverse")
histoGraph.setXTitle("S (m)")
histoGraph.setYTitle("Frequence")
histoGraph.setBoundingBox([-10,5,0,0.40])
histoGraph.setLegends([""])
View(histoGraph).show()
```

Le script précédent produit la sortie suivante :

```
Number of function calls = 100000
Failure Probability = 4.900000e-04
within [3.528361e-04,6.271639e-04]
% C.O.V: 14.3%
```

Le script produit également la figure 3, qui présente la densité des quatre variables K_s, Q, Z_m et Z_v, ainsi que la figure 4, qui présente l'histogramme de la surverse.
Bien sûr, nous aurions pu utiliser d'autres algorithmes plus avancés, comme par exemple la méthode FORM-SORM, le tirage d'importance, la stratification directionnelle classique ou adaptative ou encore la méthode Subset Simulation, puisque toutes ces méthodes sont disponibles dans OpenTURNS.

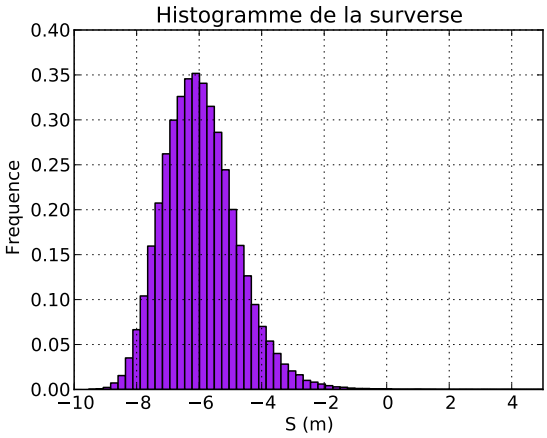

FIGURE 4 – Histogramme de la surverse.

3.8 Conclusion

Une large communauté d'acteurs industriels et académiques ont adopté la plate-forme pour réaliser leurs études ou prototyper des travaux de recherche. Cette large diffusion permet l'enrichissement de la plate-forme par de nouvelles méthodes, de plus en plus performantes et répondant à des besoins de plus en plus pointus. On peut citer, entre autres, les travaux de G. Blatman sur les polynômes de chaos [Blatman, 2009] ainsi que ceux de M. Munoz-Zuniga sur des méthodes d'estimation contrôlée de faibles probabilités [Munoz-Zuniga, 2011].

3.9 Remerciements

Nous remercions Nazih Benoumechiara (EDF R&D) de nous avoir confié un script Python implémentant le cas test présenté dans cet article.

Références

[Au and Beck, 2001] Au, S.-K. and Beck, J. L. (2001). Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic Engineering Mechanics*, 16 :263–277.

[Barate, 2013] Barate, R. (2013). Calcul haute performance avec OpenTURNS, workshop du GdR MASCOT-NUM, Quantification d'incertitude et calcul intensif". `http://www.gdr-mascotnum.fr/media/openturns-hpc-2013-03-28.pdf`.

[Blatman, 2009] Blatman, G. (2009). *Adaptive sparse polynomial chaos expansions for uncertainty propagation and sensitivity analysis*. Thèse de doctorat, Université Blaise Pascal, Clermont II, Spécialité : Génie Mécanique.

[Degoutte, 2007] Degoutte, G. (2007). Aide mémoire d'hydraulique à surface libre. `http://www.agroparistech.fr/coursenligne/hydraulique/degoutte1.pdf`.

[EADS et al., 2014a] EADS, EDF, and Phimeca Engineering (2014a). Documentation of OpenTURNS version 1.3. `http://doc.openturns.org/openturns-1.3`.

[EADS et al., 2014b] EADS, EDF, and Phimeca Engineering (2014b). OpenTURNS, a scientific library usable as a Python module dedicated to the treatment of uncertainties. `www.openturns.org`.

[H.O. Madsen, 1986] H.O. Madsen, S. Krenk, N. L. (1986). *Methods of Structural Safety*. Prentice-Hall, Englewood Cliffs, N.J.

[Munoz-Zuniga, 2011] Munoz-Zuniga, M. (2011). *Méthodes stochastiques pour l'estimation contrôlée de faibles probabilités sur des modèles physiques complexes, Application au domaine nucléaire*. Thèse de doctorat, Université Paris VII, Spécialité : Mathématiques Appliquées.

[Pasanisi and Dutfoy, 2012] Pasanisi, A. and Dutfoy, A. (2012). *An industrial viewpoint on uncertainty quantification in simulation : stakes, methods, tools, examples.*, pages 27–45. Berlin : Springer.