Université Paul Sabatier
Institut de mathématiques de Toulouse
Rapport de stage de M2R

Maîtres de stage : Fabrice Gamboa et Aurélien Garivier

Propagation d'incertitudes et estimation d'un quantile. Application à l'exposition aux ondes électromagnétiques due au téléphone cellulaire.

Tatiana Labopin-Richard

Toulouse, le 3 septembre 2013

Table des matières

1 Introduction

L'entreprise Orange souhaite avoir des informations sur le taux d'ondes éléctromagnétiques qu'absorbe un foetus lorsque sa mère utilise un téléphone portable. Différentes études ont été menées, et il est désormais connu que ce taux dépend de plusieurs paramètres (comme la position du foetus dans le ventre de sa mère). Nous considérerons dans la suite que ce nombre de paramètres est de 5 (modèle simplifié mais efficace) et que chaque paramètre est à valeurs dans $[0, 1]$ quitte à changer d'échelle.

Nous savons, à ces cinq paramètres donnés, mesurer ce taux grâce à un modèle et des machines performantes (tout cela mis en place par l'équipe de chercheurs d'Orange Labs et de Télécom Paris qui travaille actuellement sur ce problème), avec une précision quasiment parfaite (on considérera que l'erreur est nulle). Cependant, une évaluation coûte relativement cher (il faut compter une journée entière pour avoir une mesure de taux lorsque les paramètres sont fixés).

Ainsi, notre but sera de trouver une bonne estimation du quantile de la distribution du taux d'absosption, en faisant le moins possible de mesures de ce taux.

Le problème se traduit mathématiquement de la manière suivante. Nous souhaitons évaluer un paramètre d'intérêt (le quantile q_α tel que $\mathbb{P}(f(X) \geq q_\alpha) = 1-\alpha$ avec $\alpha \in]0, 1[$) de la distribution $Y = f(X)$, où X suit la loi uniforme sur $[0, 1]^d$ et représente les paramètres qui influent sur le taux d'absorption Y. f est une fonction de $[0, 1]^d$ à valeurs dans $[0, 1]$ inconnue, mais que nous savons évaluer (sans erreur) en tout point de $[0, 1]^d$ en un temps vraiment très long.

De manière plus précise, nous cherchons un ensemble de points de $[0, 1]^d$ en lesquels évaluer f, qui ne sera pas trop grand (on ne peut se permettre d'évaluer f trop de fois parce que c'est trop coûteux) mais qui nous permettra d'avoir la meilleure précision possible lors de l'estimation de notre paramètre d'intérêt.

Pour les raisons que nous avons vu plus haut, nous considérerons une dimension d de l'ordre de 5, et Orange souhaite un budget maximal de points à évaluer inférieur à 100, sachant que l'algorithme obtenu devra permettre à chaque étape de trouver le nouveau point à évaluer en moins de 1 jour.

ll apparaît rapidemment que nous avons en réalité deux problèmes a résoudre : le premier, plus facile, consiste à trouver un estimateur intéressant du paramètre d'intérêt et le deuxième, qui m'a occupée pendant la majeure partie de mon stage consiste à trouver une bonne strátégie de choix de points (*design*).

Mon stage s'est alors déroulé en deux temps. D'abord, je me suis occupée des estimateurs : j'ai cherché dans la littérature tout ce qui avait déjà été fait dans des domaines voisins (il se trouve qu'il existe beaucoup de choses sur l'estimation de la probabilité de dépasser un certain seuil et sur l'optimisation de la fonction dans ce

cadre). Cette bibliographie m'a permis de découvrir deux estimateurs couramment utilisés pour estimer la probabilité d'échec (de dépasser un certain seuil). J'ai donc dans un premier temps essayé de comprendre ces estimateurs et de les adapter à mon problème.

Dans un deuxième temps plus important, je me suis penchée sur le problème de la stratégie de choix de points. Ce problème est beaucoup plus complexe.

Il est d'abord important de remarquer que les méthodes à la Monte Carlo sont bien entendu exclues. En effet, si l'on considère par exemple dans le cadre de la probabilité d'échec :

$$\alpha_m = \frac{1}{m}\sum_{i=1}^{m} 1_{f(X_i)>u}$$

(où les X_i sont i.i.d.), on sait (par la loi des grands nombres) que α_m converge presque sûrement vers notre cible α lorsque m tend vers l'infini. L'erreur quadratique de cet estimateur vaut :

$$\mathbb{E}((\alpha_m - \alpha)^2) = \frac{1}{m}\alpha(1-\alpha)$$

Ainsi, si α est petit (ce qui est souvent le cas dans ce genre de problème), la déviation standard de α_m est de l'ordre de $\sqrt{\frac{\alpha}{m}}$. Pour atteintre une déviation standard de $\delta\alpha$, il faut donc approximativement $\frac{1}{\delta^2\alpha}$ évaluations, ce qui est grand lorsque α est petit. Par exemple (voir [**?**]), pour $\alpha = 2.10^{-3}$ et $\delta = 0.1$, on obtient $m = 50000$. Ainsi même si l'évaluation de f ne prenait qu'une minute, il faudrait 35 jours pour atteintre cette précision ! Comme dans notre cas, une évaluation prend de l'ordre d'un jour, il est clair que nous devons proscrire cette idée. Il existe alors des méthodes pour accélérer le Monte Carlo (comme l'échantillonnage préférentiel), mais la vitesse de Monte Carlo simple est tellement insuffisante que cela ne suffirait pas. Il nous faut aller chercher une méthode radicalement différente.

Dans la littérature, le type de stratégie qui semble le plus intéressant est ce qu'on apelle une stratégie d'apprentissage actif : à chaque étape, on va essayer d'utiliser au mieux l'information dont nous disposons sur la fonction grâce aux précédentes évaluations pour choisir la prochaine. Pour cela, il est habituel de se placer dans un cadre bayésien (on considère que la fonction inconnue f est la réalisation d' un processus gaussien centré de covariance connue).

J'ai alors étudié, dans ce cadre, la première méthode développée par l'équipe : la stratégie SUR. Nous verrons que cette méthode fonctionne bien mais est beaucoup trop lente. C'est pourquoi j'ai ensuite réfléchi à une nouvelle méthode (inspirée de travaux sur l'optimisation bayésienne) qui réduit considérablement le temps de calcul, comme nous le verrons à la fin de ce mémoire.

2 Processus gaussiens

2.1 Les formules de krigeage

Nous construisons, classiquement, un modèle gaussien pour la fonction f, c'est-à-dire que nous considérons que f est la réalisation d'un processus gaussien centré de fonction de covariance k connue. L'avantage de ce modèle est que nous connaissons bien la loi *a posteriori* (qui reste gaussienne), grâce aux formules de krigeage ([**?**], [**?**], [**?**]).

Propriété : formules de Krigeage

Soit Y un processus gaussien sur $\mathbb{X}$ centré de fonction de covariance définie positive k (dans la suite on prendra toujours des fonctions de covariance définies positives pour pouvoir inverser les matrices de covariances). Soit $Y_n = (Y_1, \ldots, Y_n)$ un vecteur de n observations de Y en $X_n = (x_1, \ldots, x_n)$ (n points distincts). Alors la loi de $Y(x)$ sachant la tribu $\mathcal{F}_n$ (tribu engendrée par Y_n et X_n), pour $x \in \mathbb{X}$ est gaussienne de paramètres :

$$\begin{aligned} m_n(x) &= k_n(x)^T K_n^{-1} y_n \\ c_n(x, x') &= k(x, x') - k_n(x)^T K_n^{-1} k_n(x') \end{aligned}$$

où m_n est l'espérance conditionnelle et c_n la covariance conditionnelle, $k_n(x) = [k(x_1, x), \ldots, k(x_n, x)]'$ et $K_n = [k(x_i, x_j)]_{1 \leq i,j \leq n}$. Dans la suite nous noterons $c_n(x, x) = s_n^2(x)$.

Nous remarquons alors deux choses. La première, c'est que les variances et covariances conditionnelles ne dépendent pas des évaluations de la fonction, contrairement à l'espérance conditionnelle. Ensuite, bien que ces formules soient explicites, elles peuvent être assez coûteuses lorsque n est grand puisqu'elles nécessitent l'inversion d'une matrice de taille $n \times n$. Nous allons voir comment résoudre ce problème grâce aux formules de mise à jour.

2.2 Les formules de mise à jour

Nous souhaitons donner des formules de mise à jour pour les paramètres de krigeage, c'est-à-dire des formules, qui nous permettent d'obtenir les paramètres sachant la tribu $\mathcal{F}_{n+k}$ en utilisant ceux sachant la tribu $\mathcal{F}_n$ ([**?**]). Ainsi, nous n'avons pas besoin de refaire à chaque nouveau point tout le calcul et nous évitons en particulier des inversions de matrices qui peuvent devenir de grande taille.

2.2.1 Notations

Pour établir ces formules, nous allons poser certaines notations pour alléger les calculs.

- $X_{old} = x_1, \ldots, x_n$
- $X_{new} = x_{n+1}, \ldots x_{n+k}$
- $Y_{old} = (Y(x_1), \ldots, Y(x_n))$
- $Y_{new} = (Y(x_{n+1}), \ldots, Y(x_{n+k}))$
- Pour j entier fixé, $\hat{Y}(x)_j$ et Σ_j représentent la moyenne et la variance de la loi *a posteriori* lorsque l'on a j observations.
- Pour $i \leq k$, $\lambda_{n+i|n+k}(x)$ est le poids de $Y(x_{n+i})$ dans la décomposition de la prédiction $\hat{Y}(x)_{n+k}$ suivant les $Y(x_j)$ pour $j \| eqk)$.
- $\lambda_{new,old}(x) = (\lambda_{1|n+k}(x), \ldots, \lambda_{n|n+k}(x))^T$
- $\lambda_{new,new}(x) = (\lambda_{n+1|n+k}(x), \ldots, \lambda_{n+k|n+k}(x))^T$
- Σ_{new} est la matrice de covariance de Y_{new} sachant Y_{old}.
- $\hat{Y}_{new}(x)$ sera l'espérance conditionnelle de $Y(x)$ sachant $\mathcal{F}_{new}$ (la tribu engendrée par toutes les $n+k$ observations), et de même en changeant new par old.
- On appelle toujours s^2 les variances et c les covariances indexées par leur conditionnement.

2.2.2 Les formules à k pas

Proposition : Les formules de mise à jour à k pas

$$\begin{aligned} \hat{Y}_{new}(x) &= \hat{Y}(x)_{old} + \lambda_{new,new}(x)^T (Y_{new} - \hat{Y}(X_{new})_{old}) \\ s^2_{new}(x) &= s^2_{old}(x) - \lambda_{new,new}(x)^T \Sigma_{new} \lambda_{new,new}(x) \\ c_{new}(x,y) &= s_{old}(x,y) - \lambda^T_{new,new}(x) \Sigma_{new} \lambda_{new,new}(y) \end{aligned}$$

Preuve de la première formule :

$$\begin{aligned} \hat{Y}(x)_{old} &= \mathbb{E}(Y(x)|Y_{old}) \\ &= \mathbb{E}(\mathbb{E}(Y(x)|Y_{old}, Y_{new})|Y_{old}) \\ &= \mathbb{E}(\lambda_{new,old}(x)^T Y_{old} + \lambda_{new,new}(x)^T Y_{new}|Y_{old}) \\ &= \lambda_{new,old}(x)^T Y_{old} + \lambda_{new,new}(x)^T \hat{Y}(X_{new})_{old} \\ &= \lambda_{new,old}(x)^T Y_{old} + \lambda_{new,new}(x)^T Y_{new} - \lambda_{new,new}(x)^T Y_{new} + \lambda_{new,new}(x)^T \hat{Y}(X_{new})_{old} \\ &= \hat{Y}(x)_{new} - \lambda_{new,new}(x)^T (Y_{new} - \hat{Y}(X_{new})_{old}) \end{aligned}$$

où l'on utilise seulement la définition des λ et des propriétés de l'espérance conditionnelle.

Preuve de la deuxième formule :

On procède de la même manière pour la variance.

$$\begin{aligned} s^2_{old} &= Var(Y(x)|Y_{old}) \\ &= \mathbb{E}(Var(Y(x)|Y_{old}, Y_{new})|Y_{old}) + Var(\mathbb{E}(Y(x)|Y_{old}, Y_{new})|Y_{old}) \\ &= Var(Y(x)|Y_{old}, Y_{new}) + Var(\lambda_{new,old}(x)^T Y_{old} + \lambda_{new,new}(x)^T Y_{new}|Z_{old}) \\ &= s^2_{new}(x) - \lambda_{new,new}(x)^T \Sigma_{new} \lambda_{new,new}(x) \end{aligned}$$

Preuve de la troisième formule :

C'est exactement la même idée que nous utilisons pour la troisième formule, nous ne détaillerons donc pas les calculs ici.

2.2.3 Comment trouver les λ ?

Proposition : Expression de λ

$\Sigma_{new}\lambda_{new,new}(x) = c_{old}(X_{new}, x)$

Preuve :

Nous allons utiliser l'interprétation en terme de projection orthogonale de l'espérance conditionnelle :

$$\begin{aligned} Y(x) &= \mathbb{E}(Y(x)|Y_{old}, Y_{new}) + Y(x) - \mathbb{E}(Y(x)|Y_{old}, Y_{new}) \\ &= \lambda_{new,old}(x)^T Y_{old} + \lambda_{new,new}(x)^T Y_{new} + \epsilon \end{aligned}$$

où $Y(x) - \mathbb{E}(Y(x)|Y_{old}, Y_{new}) = \epsilon$ est centré et indépendant de Y_{old} et Y_{new}.

De plus :

$$\begin{aligned} c_{old}(X_{new}, x) &:= cov(Y_{new}, Y(x)|Y_{old}) \\ &= \lambda_{new,old}(x)^T cov(Y_{new}, Y_{old}|Y_{old}) + cov(Y_{new}, \lambda_{new,new}(x)^T Y_{new}|Y_{old} \\ &\quad + cov(Y_{new}, \epsilon|Y_{old}) \\ &= 0 + \Sigma_{new}\lambda_{new,new}(x) + 0 \end{aligned}$$

Nous obtenons alors le résultat.

2.2.4 Formules à k pas simplifiées

Corollaire : Formule à k pas simplifiées

Nous avons alors directement les formules suivantes :

$$\begin{aligned} \hat{Y}(x)_{new} &= \hat{Y}(x)_{old} + c_{old}(X_{new}, x)^T \Sigma^{-1}_{new}(Y_{new} - \hat{Y}(X_{new})_{old}) \\ \sigma^2_{new}(x) &= \sigma^2_{old}(x) - c_{old}(X_{new}, x)^T \Sigma^{-1}_{new} c_{old}(X_{new}, x) \\ c_{new}(x, y) &= c_{old}(X_{new}, x)^T \Sigma^{-1}_{new} c_{old}(X_{new}, y) \end{aligned}$$

2.2.5 Les formules à un pas

Dans notre problème, nous voulons connaître les nouveaux points à évaluer un par un. Voyons donc ce que nous donnent ces formules pour $k = 1$ (avec nos notations)

Proposition : Formule de mise à jour à un pas

$$\begin{aligned} m_{n+1}(x) &= m_n(x) + \tfrac{c_n(x_{n+1},x)}{s_n^2(x_{n+1})}(Y(x_{n+1}) - m_n(x_{n+1})) \\ s_{n+1}^2(x) &= s_n^2(x) - \tfrac{c_n^2(x_{n+1},x)}{s_n^2(x_{n+1})} \\ c_{n+1}(x,y) &= c_n(x,y) - \tfrac{c_n(x_{n+1},x)c_n(x_{n+1},y)}{s_n^2(x_{n+1})} \end{aligned}$$

3 Etude de quelques estimateurs usuels

Dans cette partie, nous allons nous pencher sur notre premier problème et étudier des estimateurs pour le quantile. Nous considérons que nous avons une stratégie de choix de points fixée et nous voulons trouver un *bon* estimateur du quantile à chaque étape n, en fonction de nos précédentes observations. Nous allons d'abord étudier ce qui a déjà été fait dans le cadre de la probabilité d'échec puis nous essayerons de l'adapter à notre problème.

3.1 La probabilité de dépasser un certain seuil

Dans l'article [**?**], deux estimateurs sont proposés pour étudier la probabilité de dépasser le seuil u (on notera $\alpha(f) = \mathbb{P}(f(X) > u)$). Nous allons les donner et présenter quelques calculs que j'ai effectués pendant mon stage pour mieux comprendre les propriétés de ces deux estimateurs.

3.1.1 Deux idées d'estimateurs

Une première idée consiste à chercher, à processus gaussien ξ et stratégie fixés, l'estimateur $\hat{\alpha_n}^*$ qui minimise le risque quadratique $\mathbb{E}((\alpha - \hat{\alpha_n})^2)$ parmi tous les estimateurs $\mathcal{F}_n$-mesurables pour n compris entre 1 et n_{budget}. Si on note $\mathbb{X} = [0,1]^d$ et $P_{\mathbb{X}}$ la loi uniforme sur $[0,1]^d$, par définition de l'espérance conditionnelle ($\mathbb{E}_n$ désigne l'espérance conditionnelle par rapport à $\mathcal{F}_n$ et P_n la probabilité conditionnelle), il s'agit de :

$$\hat{\alpha_n^*} = \mathbb{E}_n(\alpha) = \mathbb{E}_n(\int_{\mathbb{X}} 1_{\xi > u} dP_{\mathbb{X}}) = \int_{\mathbb{X}} p_n dP_{\mathbb{X}}$$

où l'on a noté $p_n : x \in \mathcal{X} \mapsto P_n(\xi(x) > u) = \Phi(\frac{m_n(x) - u}{s_{n(x)}})$ et Φ la fonction de répartition gaussienne centrée réduite car ξ est un processus gaussien. Nous remarquons alors que nous pouvons avoir une approximation de cet estimateur pour tout n en utilisant un Monte Carlo pour l'intégrale et le package bivnorm du logiciel R pour les fonctions de répartition. Cependant, cela reste une approximation.

Une autre idée est assez naturelle pour construire un autre estimateur de α sachant $\mathcal{F}_n$. Il s'agit d'approcher $1_{\xi > u}$ par une autre fonction η à valeur dans $\{0,1\}$ qui en serait *proche*. Plus précisemment, on voudrait que $\hat{\alpha_n} = \int_{\mathbb{X}} \eta_n dP\mathbb{X}$ soit proche de α au sens L^2. Or, par Fubini et Cauchy-Schwatz :

$$\mathbb{E}_n((\hat{\alpha_n} - \alpha)^2) = \mathbb{E}_n[(\int (\eta_n - 1_{\xi > u}) dP_{\mathbb{X}})^2] \leq \int \mathbb{E}_n((\eta_n - 1_{\xi > u})^2) dP_{\mathbb{X}}$$

Si on pose $\tau_n(x) = \mathbb{P}_n(\eta_n(x) \neq 1_{\xi(x)>u}) = \mathbb{E}((\eta_n(x) - 1_{\xi(x)>u})^2)$, qui représente la probabilité de *mal classer le point x* , alors nous devons trouver un η tel que τ_n soit minimal pour tout x. Or en développant l'espérance on a :

$$\tau_n(x) = p_n(x) + (1 - 2p_n(x))\eta_{n(x)}$$

et donc le risque sera minimal pour :

$$\eta_n(x) = 1_{p_n(x)>0.5} = 1_{mediane(\xi_n(x))>u}$$

Mais comme la médiane et la moyenne sont égales dans notre cas :

$$\eta_n(x) = 1_{m_n(x)>u}$$

et donc :

$$\tau_n(x) = \min(p_n(x), 1-p_n(x)) = \mathbb{P}_n(\xi(x)-u)(m_n(x)-u) < 0) = 1-\Phi\left(\frac{|m_{n(x)-u}|}{s_{n(x)}}\right)$$

Finalement, nous tombons sur $\hat{\alpha_n} = \alpha(m_n)$, estimateur de type plug-in.

Les deux estimateurs proposés sont alors faciles à comprendre. Le premier calcule la probabilité que le processus gaussien conditionnellement aux points déjà évalués soit supérieur à u et en prend la moyenne. Le deuxième calcule les moyennes et regarde la probabilité qu'elles soient au-dessus du seuil u.

3.1.2 Comparaison des estimateurs

Ces estimateurs, bien qu'assez intuitifs, sont difficilement manipulables parce qu'ils font intervenir de gros calculs. S'il est à peu près clair que lorsque nous générons des points qui remplissent uniformément l'espace, et sous des hypothèses de régularité de la fonction, ces estimateurs sont consistants (nous verrons un exemple plus loin), cette propriété n'est pas évidente lorsque nous jouons des points grâce à une stratégie différente.

Pour mieux les comprendre, nous voulions d'abord savoir si l'un des deux estimateurs était meilleur que l'autre. L'estimateur de type plug-in semble plus simple à manipuler, d'autant plus qu'il est assez clair que l'autre estimateur est calculé par Monte Carlo. Cependant, nous avions l'impression que l'estimateur "le plus compliqué" était le plus intéressant. Malheureusement, nous n'avons pas réussi à bien formaliser cette idée. Nous allons présenter l'analyse que nous avons fait en dimension 1. Pour commencer, il apparaît rapidemment, que dans le cas général, aucun des deux estimateurs n'est meilleur que l'autre. En effet, si on prend un u

tel que $\forall x \in [0,1]$, $m_n(x) > u$, $\hat{\alpha}_n^1 = 1$ alors que $\hat{\alpha}_n^2 < 1$. Au contraire, si nous prenons un u plus grand en tout point x que m_n alors le premier estimateur est nul alors que le deuxième est strictement positif.

Nous nous sommes alors demandés si nous pouvions exhiber une inégalité entre ces deux estimateurs. Nous pouvons alors montré que :

Propriété :

$$\frac{1}{2}\hat{\alpha}_n^1 \leq \hat{\alpha}_n^2 \leq \frac{1}{2} + \frac{1}{2}\hat{\alpha}_n^1$$

Preuve :

Il suffit de remarquer que lorsque nous avons un x tel que $m_n(x) > u$ alors $\hat{\alpha}_n^1 = 1$ et $\frac{1}{2} < \hat{\alpha}_n^2 = \Phi(\frac{m_n(x-u}{s_{n(x)}}) < 1$ et que si nous avons un x tel que $m_n(x) < u$ alors $\hat{\alpha}_n^1 = 0$ et $\frac{1}{2} > \hat{\alpha}_n^2 > 0$ (puisque l'ensemble des x tel que $m_n(x) = u$ est de mesure nulle). Ainsi, en intégrant sur $[0,1]$, puis en coupant l'intégrale en deux, nous trouvons directement le résultat (on rappelle qu'on intégre selon la loi uniforme).

Malheureusement, n'ayant aucune hypothèse sur m_n et s_n, nous ne pouvons affiner cette inégalité. On remarque de plus, que les deux cas limites exposés au début de la partie sont bien les cas d'égalité de nos inégalités.

Pour rendre ces estimateurs plus manipulables et essayer d'aller plus loin dans les calculs, nous avons étudié les estimateurs précédents dans un cadre précis : celui où la covariance est brownienne.

3.1.3 Calculs explicites dans le cas de la covariance brownienne

Nous avons ensuite essayé de faire les calculs explicites des estimateurs dans le cas de la covariance brownienne (toujours en dimension 1). D'abord, j'ai regardé la loi *a posteriori* de ce modèle. Lorsque nous ordonnons les points d'observations (x_i), la matrice de covariance est de la forme suivante :

$$\begin{pmatrix} x_1 & x_1 & x_1 & ... & x_1 \\ x_1 & x_2 & x_2 & ... & x_2 \\ x_1 & x_2 & x_3 & ... & x_3 \\ ... & ... & ... & ... & ... \\ x_1 & x_2 & x_3 & ... & x_n \end{pmatrix}$$

Or cette matrice à un inverse d'une forme simple : elle est tri-diagonale. Ainsi, comme c'est l'inverse de cette matrice qui apparaît dans la formule de la densité, nous avons un caractère markovien : $\mathbb{E}(\xi(x)|\xi(x_1,\ldots,\xi(x_n)) = \mathbb{E}(\xi(x)|\xi(x_i),\xi(x_{i+1}))$

où $x_1 < \cdots < x_n$ et $x_i \leq x \leq x_{i+1}$. Donc, nous remarquons que l'estimée de f au point x ne dépend que des deux points qui l'entourent qui ont été évalués (et que si x n'a qu'un plus grand point évalué ou qu'un plus petit point évalué alors c'est ce seul point qui influe). Cela m'a permis de trouver la formule pour n points évalués, en ne faisant les calculs que dans le cas de deux points évalués et en découpant la partition.

Dans la suite, on notera donc x_1 le point évalué inférieur à x et x_2 le point évalué supérieur à x si les deux points existent. Une application directe des formules de krigeage données dans les parties précédentes donnent :

$$\begin{aligned} m_n(x) &= \frac{f(x_1)(x_2-x)+f(x_2)(x-x_1)}{x_2-x_1} \\ s_n(x) &= \sqrt{\frac{(x_2-x)(x-x_1)}{x_2-x_1}} \end{aligned}$$

De plus m_n est constante à la première valeur évaluée à gauche du plus petit point évalué et constante égale à la dernière à droite de la plus grande. On reconnait donc l'interpolation linéaire classique. Calculons maintenant les deux estimateurs précédents.

Le premier est très simple on trace l'interpolation linéaire des points évalués et on regarde lorsque cette courbe est au dessus de la droite $y = u$. On peut écrire de manière explicite la forme de cet estimateur (pour le programmer par exemple). Pour cela on ordonne les points d'évaluation par ordre croissant et on a :

$$\begin{aligned} \hat{\alpha}_n^1 &= \sum_{i=1}^{n-1} [1_{f(x_{i+1})-f(x_i)>0}(1_{(f(x_i)<u<f(x_{i+1})}(\frac{x_{i+1}-x_i)(f(x_{i+1})-u)}{f(x_{i+1})-f(x_i)}) + (x_{i+1}-x_i)1_{f(x_i)>u}1_{f(x_{i+1})>u}] \\ &+ [1_{f(x_{i+1})-f(x_i)<0}(1_{(f(x_i)>u>f(x_{i+1})}(\frac{x_{i+1}-x_i)(u-f(x_i)}{f(x_{i+1})-f(x_i)}) + (x_{i+1}-x_i)1_{f(x_i)>u}1_{f(x_{i+1})>u}] \\ &+ (x_1 - \frac{x_1 u}{f(x_1)})1_{f(x_1)>u}) + (1-x_n)1_{f(x_n)>u}) \end{aligned}$$

Le deuxième reste compliqué. Considérons pour éliminer les problèmes bords que nous évaluons la fonction en 0 et en 1. Nous avons alors :

$$\frac{m_{n,i}(x)-u}{s_{n,i}(x)} = \frac{f(x_1)(x_2-x)+f(x_2)(x-x_1)-u(x_2-x_1)}{\sqrt{(x_2-x)(x-x_1)(x_2-x_1)}}$$

$$\begin{aligned} \hat{\alpha}_n^1 &= \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} \Phi(\frac{m_{n,i}(x)-u}{s_{n,i}(x)})dx \\ &= \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} \int_{-\infty}^{\frac{f(x_1)(x_2-x)+f(x_2)(x-x_1)-u(x_2-x_1)}{\sqrt{(x_2-x)(x-x_1)(x_2-x_1)}}} \exp(-\frac{v^2}{2})\frac{dv}{\sqrt{2\pi}}dx \end{aligned}$$

Nous n'avons jusqu'alors trouvé aucun moyen de simplifier cette intégrale : aucun changement de variable ne nous permet de la calculer directement à cause de la racine carrée ; utiliser l'équivalent de la fonction de répartition ne nous aide pas plus (il nous permet seulement de retomber sur l'inégalité que l'on connait dans le cas où u est plus grand ou plus petit que $m_n(x)$) et une intégration par partie n'est pas possible.

Finalement, nous avons essayé d'affiner l'inégalité entre les deux estimateurs que nous avions obtenue dans le cas général. Pour cela, nous avons voulu comparer deux petites contributions des estimateurs (pas toute la somme des intégrales mais juste un morceau de la somme, puisqu'ensuite il suffisait de sommer). Comme nous venons de le voir, le calcul est très difficile, nous avons donc commencé à raisonner seulement dans le cas où $f(x_1) = f(x_2)$. Je ne présente ici que l'idée. On considère que les x_i sont ordonnés. Dans ce cas, nous avons :

$$\hat{\alpha}^2_{n,1,2} = \int_{x_1}^{x_2} \int_{-\infty}^{\frac{(f(x_1)-u)\sqrt{x_2-x_1}}{\sqrt{(x_2-x)(x-x_1)}}} \exp(-\frac{-v^2}{2})\frac{dv}{\sqrt{2\pi}}dx$$

Ainsi, dès que x_1 est plus grand que u, nous pouvons utiliser l'approximation $\Phi(u) \approx 1 - \frac{\phi(u)}{u}$ valide si u est assez grand :

$$\hat{\alpha}^2_{n,1,2} \approx (x_2 - x_1) - \int_{x_1}^{x_2} \exp(-\frac{(f(x_1)-u)^2(x_2-x_1)}{2[(x_2-x)(x-x_1)]^2})\frac{\sqrt{(x_2-x)(x-x_1)}}{(f(x_1)-u)\sqrt{x_2-x_1}}\frac{\sqrt{2\pi}}{dx}$$

Or, en faisant l'étude de la fonction $x \mapsto \exp(-\frac{(f(x_1)-u)^2(x_2-x_1)}{2[(x_2-x)(x-x_1)]^2})\frac{\sqrt{(x_2-x)(x-x_1)}}{(f(x_1)-u)\sqrt{x_2-x_1}}\frac{1}{\sqrt{2\pi}}$, on trouve que $f(x) \leq f(\frac{x_1+x_2}{2})$ et $f(x) \geq f(x_1) = 0 \ \forall x \in [x_1, x_2]$. Ainsi, si $f(x_1) = f(x_2) >> u$ (assez pour que l'approximation soit cohérente), nous avons :

$$\hat{\alpha}^2_{n,1,2} = (x_2 - x_1) - \int_{x_1}^{x_2} f(x)dx \geq (x_2 - x_1) - (x_2 - x_1)f(\frac{x_1 + x_2}{2})$$

et :

$$\hat{\alpha}^2_{n,1,2} = (x_2 - x_1) - \int_{x_1}^{x_2} f(x)dx \| eq (x_2 - x_1) - 0$$

Or nous savons que sous les mêmes hypothèses $\hat{\alpha}^1_{n,1,2} = (x_2 - x_1)$. Nous trouvons donc une autre inégalité un peu plus précise dans ce cas là :

$$\hat{\alpha}^1_{n,1,2}(x_2 - x_1)(1 - \exp(\frac{(f(x_1)-u)^2}{4(x_2-x_1)})\frac{\sqrt{x_2-x_1}}{2\sqrt{2\pi}(f(x_1)-u)} \leq \hat{\alpha}^2_{n,1,2} \leq \hat{\alpha}^1_{n,1,2}$$

De la même manière, si $f(x_1) = f(x_2) << u$:

$$\hat{\alpha}^1_{n,1,2} \leq \hat{\alpha}^2_{n,1,2} \leq \hat{\alpha}^1_{n,1,2}(x_2 - x_1)(1 + \exp(\frac{(f(x_1) - u)^2}{4(x_2 - x_1)})\frac{\sqrt{x_2 - x_1}}{2\sqrt{2\pi}(u - f(x_1))})$$

Cependant, si (x_1) et $f(x_2)$ sont égaux et vraiment trop proches de u, alors on ne peut pas conclure. De plus, lorsque nous avons ensuite essayé d'utiliser ces résultats pour voir ce qu'il se passait lorsque $f(x_1) \neq f(x_2)$, en dégénérant peu à peu, cela devenait vraiment trop compliqué, nous avons préféré passer à autre chose.

3.1.4 Discussion sur la consistance de l'estimateur 1 dans le cas de la covariance brownienne

Voyant la simplicité de l'estimateur numéro 1 dans le cas de la covariance brownienne, nous avons voulu regarder si le premier estimateur était consistant (au moins dans le cas où les (x_i) finissent par remplir [0, 1] uniformément). L'idée de notre réflexion fut la suivante : nous savons que sous certaines hypothèses, l'interpolation linéaire converge uniformément vers la fonction cible. Nous voulons donc voir si cette propriété de convergence se transfert à l'estimateur $\hat{\alpha}^2_n$:

Propriété : Convergence de l'interpolation linéaire

On considère une fonction à valeurs réelles f définie sur $[0, 1]$ et pour toute partition de n points de $[0, 1]$ ordonnés (avec $x_1 = 0$ et $x_n = 1$) qu'on note $(x_i)_n$, on pose

$$P_{i,n}(.) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(. - x_i),\ P_n(x) = \sum_{i=1}^{n} P_i(x)1_{x\in[x_i,x_{i+1}]}$$

Si on considère des partitions $(x^n_i)_n$ qui remplissent uniformément $[0, 1]$ lorsque n tend vers l'infini (c'est à dire que pour tout $h > 0$, il existe un entier n et une partition $(x^n_i)_n$ de $[0, 1]$ de pas constant h), et si f est k- lipschitzienne, alors P_n converge uniformément vers f lorsque n tend vers l'infini.

Preuve :

Soit $\epsilon > 0$. Comme f est k-lipschitzienne sur le compact $[0, 1]$, elle est uniformément continue sur ce compact. Soit ω le module de continuité de f, nous savons donc que $\omega(\delta)$ tend vers 0 lorsque δ tend vers 0. Ainsi, $\exists h > 0$ tel que $\omega(h) \leq \epsilon$. Soit n et $(x^n_i)_n$ la partition de $[0, 1]$ de pas constant égal à h et soit $x \in [0, 1]$. Alors :

$$|f(x) - P_n(x)| \leq \omega(h) \leq \epsilon$$

Remarque : Nous nous sommes ici penchés sur le cas lipschitzien parce que c'est une régularité assez faible (comparée au théorème habituel de convergence uniforme qui nécessite une fonction $\mathcal{C}^2$ mais surtout parce que lorsque nous chercherons une stratégie de choix de points, il pourra être intéressant (ce n'est encore qu'une idée parce que nous n'avons pas utilisé ce genre de choses pendant le stage) de faire l'hypothèse que la fonction f est lipchitzienne. En effet, dans ce cas, lorsque l'on évalue f en un point x_i, on connait finalement f dans tout un voisinage $B(x_i, \frac{|f(x_i)-u|}{k})$, ce qui semble interessant lorsque l'on ne veut pas faire d'exploration inutile.

Consistance de l'estimateur

Nous voulons savoir si dans le cas où les points d'évaluation finissent par remplir uniformément $[0, 1]$ le premier estimateur est consistant. Nous espérons donc montrer que :

$$\int_0^1 |1_{m_{n(x)}>u} - 1_{f(x)>u}|dx \mapsto_{n\mapsto\infty} 0$$

Comme nous l'avons déjà évoqué, nous ne pouvons pas passer directement à la limite dans l'intégrale par manque de continuité. En réalité, tout se passe bien lorsque nous sommes loin de u et là où l'on se rapproche de u, la discontinuité pose un petit problème. Pour le régler, nous allons faire une hypothèse qui s'avérera faible et suffisante. Soit $\eta > 0$, on suppose qu'il existe $\epsilon_\eta / \mu(A_\epsilon) \leq \eta$ où $A_\epsilon = \{x, f(x) \in [u-\epsilon, u+\epsilon]\}$. Soit $N_{\epsilon_\eta} / \forall x, \forall n \geq N_{\epsilon_\eta} |m_n(x) - f(x)| < \epsilon_\eta$. Alors :

$$\int_0^1 |1_{m_{n(x)>u}} - 1_{f(x)>u}|dx = \int_0^1 |1_{m_{n(x)>u>f(x)}} - 1_{f(x)>u>m_n(x)}|dx$$

D'après notre hypothèse, nous avons $-\epsilon + f(x) < m_{n(x)} < \epsilon + f(x)$, ainsi, si x est tel que $f(x) > u + \epsilon$ alors $m_{n(x)} > u$ et si x est tel que $f(x) < u - \epsilon$ alors $m_{n(x)} < u$. Dans ces deux cas là, les indicatrices s'annulent (et c'est cohérent puisque c'est en u que le problème de continuité intervient). Finalement :

$$\int_0^1 |1_{m_{n(x)>u}} - 1_{f(x)>u}|dx \leq \int_0^1 1_{f(x)\in[u-\epsilon,u+\epsilon]}dx \leq \mu(A_\epsilon) \leq \eta$$

où l'on a grossièrement majoré la différence d'indicatrice par 1.

Il nous reste donc à caractériser les fonctions pour lesquelles, pour tout ϵ, il existe un tel η_ϵ. Les fonctions qui sont à un moment constantes égales à u ne vérifient bien entendu pas cette propriété. Les fonctions qui coupent $y = u$ une infinité de fois de manière générale peuvent ne pas vérifier l'hypothèse. Mais grâce à la compacité de $[0, 1]$, nous pouvons encore restreindre l'ensemble des fonctions qui ne fonctionnent pas. En effet, si on considère la suite des points où f coupe $y = u$ alors, comme elle vit dans le compact [0, 1], elle admet au moins un

point d'accumulation. Si cet ensemble de points d'accumulation est fini, on arrive tout de même à vérifier l'hypothèse. Il faut donc seulement enlever les fonctions coupant $y = u$ une infinité de fois et dont les tels points d'accumulation sont infinis. Finalement, il apparaît que la consistance est vraie dès lors que f est un tout petit peu régulière.

3.1.5 Tentative de généralisation de l'estimateur 1 dans le cas de la covariance brownienne

Voyant la simplicité de l'esimateur 1 en dimension 1 avec covariance brownienne, nous nous sommes demandés si nous pouvions trouver une généralisation de cet estimateur en dimension supérieure. On rappelle le fonctionnement du premier estimateur en dimension 1 : on trace l'interpolation linéaire de la fonction grâce aux points évalués puis nous évaluons la proportion (nous sommes sur $[0, 1]$ avec une loi uniforme) où cette fonction est au dessus de u en sommant la longueur des segments qui correspondent à ces zones qu'on a projeté sur l'axe des abscisses.

Si on se place en dimension $d = 2$, on peut se dire qu'une généralisation logique serait de passer d'une contribution sous la forme d'un segment à celle sous la forme d'une aire. Ainsi, nous avons d'abord pensé qu'on pourrait, une fois que n points sont évalués, considérer les points trois par trois, regarder le triangle plein qu'ils forment et projeter la partie de ce triangle qui est au dessus du plan $y = u$ sur le plan des abscisses. L'aire de cette projection nous donnerait alors la contribution des trois points considérés à l'estimation de notre paramètre. Cette méthode semble à première vue réalisable parce que tout est possible à mettre en équation grâce à de la géométrie élémentaire dans l'espace. Mais en y regardant de plus près, nous nous sommes alors rendu compte d'un problème : en dimension 1, lorsque l'on ajoutait un point, il se trouvait entre deux points déjà évalués seulement et donc ajouter un point, nous faisait seulement remplacer un segment par deux segments. Ici, ça semble beaucoup plus compliqué. Si on fait le calcul, on se rend compte que lorsqu'on à n points évalués et qu'on en ajoute un, nous avons $\binom{n}{2}$ nouveaux triangles qui apparaissent. On peut facilement voir que cette méthode peut se généraliser à l'identique pour d encore plus grand mais la complexité exponentielle n'est pas appropriée.

Nous avons alors réfléchi à une autre manière de procéder et avons pensé à une première simplification. Pour calquer de la dimension 1, l'idée d'aller chercher seulement les deux points les plus proches de x, on peut se dire en dimension 2, que nous allons chercher le quadrilatère le "plus proche" de x en regardant les deux points d'abscisses immédiatemment plus petites et plus grandes que x et les deux points d'ordonnées immédiatemment plus grandes et plus petites que x. Ainsi,

lorsque l'on ajoute un point et qu'on le considère dans son quadrilatère le plus proche, nous n'ajoutons que $\binom{2^2}{2}$ triangles. Cette méthode se généralise encore une fois à la dimension supérieure : en dimension d, on ajoute à chaque point $\binom{2^d}{d}$ nouveaux polyèdres. La complexité reste encore trop élevée et nous avons eu une dernière idée pour la réduire.

Nous avons voulu utiliser la triangulation : on regarde le triangle "le plus proche de x" et non le "quadrilatère". Si nous triangulons notre espace, chaque nouveau point va nous donner au plus 3 nouveaux triangles. La complexité est cette fois linéaire mais d'une part nous ne savons pas comment généraliser ça en dimension plus grande que 2 et d'autre part, il nous paraissait vraiment trop compliqué de programmer cela. Nous avons donc arrêté l'étude ici.

3.2 Le quantile

L'estimateur du quantile classique est le quantile empirique. Pour commencer nous allons faire des rappels sur les propriétés de cet estimateur. Ensuite nous verrons comment à partir de ce quantile, adapter les deux estimateurs précédents à notre problème.

3.2.1 Quantile empirique

Nous définissons un quantile d'ordre α de la manière suivante. On appelle F la fonction de répartition de la loi Y . Alors un quantile d'ordre $\alpha \in [0, 1]$ est un réel q_α tel que $\mathbb{P}(Y \geq q_\alpha) \leq 1 - \alpha$.

Où l'on prendra α compris entre 0 et 1 proche de 1. Nous voulons par exemple trouver le niveau tel que Y sera au dessus au plus 5 % du temps (ici $\alpha = 95$ %).

Pour la suite et pour avoir les bonnes propriétés sur le quantile empirique, nous allons considérer que nous sommes dans le cas régulier, où l'inverse généralisée de F qu'on note F^{-1} est continue, c'est à dire que pour tout α il y a un unique quantile d'ordre α et que ce quantile vérifie $F(q_\alpha) = \alpha$. Finalement, nous cherchons à estimer le nombre q_α tel que $\mathbb{P}(f(X) \geq q_\alpha) = 1 - \alpha$.

Le quantile empirique se construit de la manière suivante : on trie nos n observations par ordre croissant (on obtient $y_{(1)}, \ldots, y_{(n)}$) et on fait en sorte de prendre l'observation telle qu'il y ait au plus 5 % des observations plus grandes, cela revient à choisir $y_{([n\alpha]+1)}$.

Consistance forte du quantile empirique :

Théorème :
0.1cm
Soit $\alpha \in [0,1]$. Sous nos hypothèses, le quantile empirique converge presque sûrement vers le quantile lorsque le nombre d'observations tend vers l'infini.

Preuve :

C'est une application du théorème de Glivenko-Cantelli. Appelons F_n la fonction de répartition empirique.

$$|F(Y_{([n\alpha]+1)}) - F(q_\alpha)| \leq |F(Y_{([n\alpha]+1)}) - F_n(Y_{([n\alpha]+1)})| + |F_n(Y_{([n\alpha]+1)}) - F(q_\alpha)|$$

Or, par Glivenko-Cantelli, presque sûrement :

$$|F(Y_{([n\alpha]+1)}) - F_n Y_{([n\alpha]+1)}| \leq Sup_x |F(x) - F_n(x)| \mapsto_{n\mapsto+\infty} 0$$

et

$$|F_n(Y_{([n\alpha]+1)}) - F(q_\alpha)| = |\frac{[n\alpha]+1}{n} - \alpha| \mapsto_{n\mapsto+\infty} 0$$

Ainsi,
$F(Y_{([n\alpha]+1)})$ converge presque sûrement vers $F(q_\alpha) = \alpha$.
Comme par hypothèse, F^{-1} est continue,on en déduit que $Y_{([n\alpha]+1)}$ converge presque sûrement vers $F^{-1}(\alpha) = q_\alpha$, d'où la consistance forte.

Théorème : Un TCL pour le quantile empirique

Soit $\alpha \in]0,1[$ et supposons que F a pour densité f strictement positive au voisinnage de q_α alors :

$$\sqrt{n}(Y_{([n\alpha]+1)}) - q_\alpha) \mapsto_{\mathcal{L}} \mathcal{N}(0, \sigma_\alpha^2)$$

avec $\sigma_\alpha^2 = \frac{\alpha(1-\alpha)}{f(q_\alpha)^2}$.

Preuve :

Elle utilise la loi des statistiques d'ordre, le théorème de Donsker et la delta méthode. Nous ne la détaillerons pas ici, on peut la trouver dans [**?**].

3.2.2 Deux estimateurs du quantile pour notre problème

Nous voulons étudier le quantile de la distribution $Y = f(X)$. Si nous générons des points x uniformément, le quantile empirique est alors un bon estimateur du quantile. En revanche, il est important de comprendre que lorsque nous allons générer des points x qui ne seront plus uniformément répartis mais générés selon

une stratégie, nous ne pourrons plus appliquer le quantile empirique au vecteur des observations, parce que nous aurons modifié la loi en forçant les points d'évaluation à se trouver à tel ou tel endroit. C'est pourquoi nous allons essayer d'adapter les estimateurs de la probabilité d'échec, vus dans la partie précédente, à notre problème.

On appelle $q(f)$ le quantile d'ordre α de la distribution $f(X)$. Les deux estimateurs précédemment proposés se généralisent facilement en :

- $\hat{q}_n^1 = \mathbb{E}_n[q(Y)]$
- $\hat{q}_n^2 = q(m_n)$

En revanche, le quantile d'une fonction de X n'a pas en générale une forme analytique. Nous allons donc l'approcher par le quantile empirique. Finalement, en notant le quantile empirique $\hat{q}_n^*$, nos deux estimateurs sont les suivants :

- $\hat{q}_n^1 = \mathbb{E}_n[\hat{q}_n^*(Y)]$
- $\hat{q}_n^2 = \hat{q}_n^*(m_n)$

Maintenant que nous avons choisi nos estimateurs, nous allons réfléchir au problème plus compliqué du choix des points à évaluer.

4 Quel genre de stratégie ?

Nous nous penchons à présent sur le deuxième problème : la stratégie de choix du plan d'expérience. Il y a deux types de stratégie qui peuvent venir à l'esprit. La première consiste à déterminer d'abord tous les points à évaluer puis à les évaluer seulement ensuite. C'est ce qu'on appelle une stratégie batch. La deuxième consiste à construire les points au fur à mesure, en utilisant ce que nous connaissons déjà de la fonction à chaque étape, c'est ce qu'on appelle de l'apprentissage actif.

4.1 Des méthodes batch à l'apprentissage actif

La méthode la plus basique consiste à ajouter à chaque étape un nouveau point tiré au hasard. Nous n'utilisons alors pas l'information que nous avons déjà à l'instant n, et on peut donc choisir tous les points avant de lancer une évaluation. Les articles qui ont testé cette méthode (par exemple [**?**]), montre toujours qu'on peut facilement trouver mieux. On peut aisémment comprendre pourquoi : notre problème n'est pas de connaître f parfaitement partout, mais de trouver son quantile. Ainsi, nous souhaiterions générer plus de points là où f prend ses plus grandes valeurs et peu de points là où elle prend ses plus petites valeurs. Il semble ainsi pertinent d'utiliser à chaque étape, toute l'information disponible sur f grâce aux précédentes évaluations, pour ne pas générer de points inutiles, tout en faisant attention à ne pas oublier de zones intéressantes. En résumé, nous voulons une stratégie d'apprentissage actif permettant de faire un compromis entre exploration (il ne faut pas oublier de zones intéressantes) et exploitation (il ne faut pas générer de points inutiles). On va donc dans la suite, chercher des méthodes d'apprentissage actif qui se présenterons selon les étapes suivantes :

1) On fabrique une grille de l'espace $[0,1]^d$ que l'on notera $\mathcal{X}$.
2) On choisit indépendemment de f une sous-grille de $\mathcal{X}$ que l'on notera $\mathcal{X}_0$ (de taille n_0) et qui constituera les points que l'on évaluera pour initialiser notre algorithme. Après ces évaluations, on calcule l'estimateur de notre paramètre d'intérêt sous notre modèle conditionnellement au n_0 évaluations, on le note $\theta_{n_0}(f)$. On se retrouve donc avec une tribu $\mathcal{F}_0$ engendrée par $\mathcal{X}_0$, leurs images par f et l'estimateur courant.
3) A chaque étape n (en partant de 1 et en allant jusqu'à ce que notre budget soit atteint) on cherche un nouveau point x_n à évaluer, en utilisant seulement la tribu $\mathcal{F}_{n-1}$. On évalue notre nouveau point et on met à jour notre estimateur.

L'étape 1 peut se faire de manière très efficace en utilisant le package LHS du logiciel R et la fonction randomLHS. La stratégie de choix de la boucle de l'étape 3 est évidemment la partie la plus délicate et c'est ce que nous cherchons

à déterminer.

Le problème de choisir le prochain point *optimal* n'est pas un problème simple. On peut d'abord penser qu'une bonne méthode serait de diminuer l'incertitude que l'on a sur f (c'est à dire sa variance conditionnelle) à chaque étape, c'est ce qu'on apelle la méthode GPExplor (exploration pure). Ainsi :

$$x_{n+1} = Argmax_{x \in \mathcal{X}} s_n^2(x)$$

Mais il apparaît vite que cette stratégie n'est pas optimale (voir [**?**] par exemple). En effet, ici, nous n'utilisons pas réellement toute l'information donnée par la tribu de l'étape précédente parce que comme nous l'avons remarqué au début de ce mémoire, la variance conditionnelle ne dépend pas des évaluations. Cette stratégie est en réalité encore une stratégie d'exploration pure, elle est un peu équivalente à jouer une répartition de points avec la fonction LHS. Nous recherchons donc une méthode d'apprentissage actif, qui utilisera au maximum la tribu de l'étape n lors du choix du $(n+1)$ième point, et qui pourra grâce à cela faire un compromis entre exploration et exploitation.

5 Choix de la stratégie SUR : la première stratégie de l'équipe

5.1 Explication de la stratégie

L'équipe Télécom-Orange Labs a décidé d'utiliser la stratégie d'apprentissage actif suivante : la méthode SUR (voir [**?**]). L'idée de cette méthode est de minimiser en moyenne la variance de l'estimateur courant (Stepwise Uncertainty Reduction). Plus précisemment, lorsque nous avons n évaluations, nous allons chercher sur la grille $\mathcal{X}$ le point suivant :

$$x_{n+1} = \min_{x \ in \mathcal{X}} V_n(x)$$

où :

$$V_n(x) = \int \mathrm{Var}(\theta(f)|\mathcal{F}_n^{(x,y)})\phi_{(m_{n(x)},s_n^2(x))}(y)dy$$

où ϕ est la densité gaussienne.

Ainsi à l'étape n, pour chaque point de la grille, on évalue quelle serait la variance de l'estimateur du quantile courant à l'étape $(n+1)$ si on prenait ce point comme $(n+1)$ième point. Malheureusement, pour calculer cette variance, nous avons besoin de l'évaluation de f en ce point, ce que nous ne pouvons obtenir (nous cherchons si nous voulons ou non évaluer ce point, il serait donc insensé de l'évaluer pour se décider...). Nous utilisons alors, pour combler ce manque d'information, la seule chose que nous connaissons : la loi de $f(X)$ sachant $\mathcal{F}_n$. Nous intégrons selon cette loi, ce qui nous donne le critère ci-dessus. On remarque que ce critère est indexé par n et x alors qu'il représente quelque chose calculable à l'étape $(n+1)$. Mais tout l'intérêt de la méthode est bien entendu de le rendre calculable avec le tribu $\mathcal{F}_n$ et x seulement !

Cette technique semble plutôt bonne en théorie mais il vient rapidemment un problème de complexité de calcul. Le critère n'a pas de forme analytique ; L'intégration suivant la loi gaussiennne ne simplifie pas les choses et nécessite une approximation Monte Carlo.

5.2 Estimation du quantile par la stratégie SUR

Voici l'algorithme retenu par l'équipe pour la méthode SUR. Comme nous l'avons déjà remarqué, cet algorithme nécessite beaucoup d'approximations Monte Carlo. On peut constater 3 boucles Monte Carlo imbriquées pour l'estimation du critère et sa minimisation.

Entrées : une grille $\mathcal{X}$ finie de $[0,1]^d$, n_0 un petit nombre et $\mathcal{X}_0$ les n_0 points de $\mathcal{X}$ où l'on évalue f pour initialiser (on notera toujours $\mathcal{F}_{n_0}$ la tribu engendrée par $((X_1, f(X_1)), \ldots, (X_{n_0}, f(X_{n_0})))$. On notera ξ_n le processus f sachant $\mathcal{F}_n$.

Algorithme :

1. Evaluer la distribution *a posteriori* ξ_{n_0} par les formules de krigeage ($\xi_{n_0} \sim PG(\mu_{n_0}, k_{n_0})$).
2. Estimer le quantile dans ce modèle substitut grâce aux estimateurs étudiés dans les parties précédentes.
3. Simuler un N-échantillon de ξ_{n_0} : $(\xi_{n_0}^{(1)}, \xi_{n_0}^{(2)}, \ldots, \xi_{n_0}^{(N)})$.
4. Estimation du critère pour tout $x \in \mathcal{X}/\mathcal{X}_0$: boucle sur les x.
   A. Pour i allant de 1 à N :
      a. Evaluer la distribution postérieure de $\xi_{n_0}^{(x,\xi_{n0}^i(x))}$.
      b. Simuler un N'-échantillon $(\xi^{(1)}_{n_0^{(x,\xi_{n0}^i(x))}}, \ldots, \xi^{(N')}_{n_0^{(x,\xi_{n0}^i(x))}})$.
      c. Pour j allant de 1 à N' :
         a'.) Estimer le quantile $\theta(\xi^{(j)}_{n_0^{(x,\xi_{n0}^i(x))}})$.
      d. Calculer la variance empirique des θ : $s_n^{(i)}(x)$.
   B. Evaluer $V_n(x) = \frac{1}{N}\sum_{i=1}^{N} s_{n^{(i)}}(x)$.
5. Minimiser sur x le critère $V_n(x)$.
6. Evaluer le point qui minimise et mettre à jour les observations.
7. Retourner à la première étape si le budget d'évaluations n'est pas atteint.

Sortie : Estimateur du quantile avec n_{budget} évaluations.

Nous verrons que les résulats de cette méthode en dimension 2 sont bons. Mais il est en revanche très difficile d'utiliser cet algorithme en dimension supérieure à cause de la complexité de calcul. Nous objectif est alors de trouver un algorithme qui donne d'aussi bons résultats mais qui a une plus faible complexité de calcul (on veut notamment supprimer les boucles Monte Carlo). Le programme R est disponible en annexe.

6 Une nouvelle méthode : le critère SUR revisité

6.1 Idée

On souhaite donc construire pas à pas une suite de points sur lesquels on évaluera f. Nous voulons que cette suite nous permette de connaître à chaque étape, au mieux, l'information que nous recherchons sur f. Le principe du critère SUR, comme nous l'avons vu précédemment est alors de choisir pour l'étape $(n+1)$, le point qui minimisera un certain critère représentant l'incertitude au temps $(n+1)$. C'est ainsi que nous étions plutôt ramené à minimiser une variance. Dans cette partie, nous allons changer un tout petit peu de point de vue, et utiliser non pas un critère quantifiant l'incertitude de notre modèle mais plutôt mesurant en quelque sorte l'erreur que nous commettons avec notre estimateur courant.

Cette idée a déjà été utilisée pour l'estimation du minimum d'une fonction (voir [**?**]), dans le même cadre que le notre. Le critère retenu pour ce problème est alors le suivant :

$$\Gamma_n = \int_{\mathbb{X}} \mathbb{P}_n(Y(x) \leq y^n_{min})dx$$

où y^n_{min} représente l'estimateur du minimum au temps n. L'idée est donc bien de regarder, en moyenne, la probabilité de se tromper.

La stratégie est ensuite la suivante : au début de l'étape "$n+1$", on cherche le point d'évaluation x_{n+1} qui va minimiser le critère de l'étape $(n+1)$. En fait on recherche le point tel que si on le choisit à l'étape $(n+1)$, on aura en moyenne la plus petite probabilité de se tromper :

$$x_{n+1} = \underset{\mathbb{X}}{\text{argmin}}\Gamma_{n+1} = \underset{\mathbb{X}}{\text{argmin}} \int_{\mathbb{X}} \mathbb{P}_n\Big(Y(x) \leq \min(y^n_{min}, y_{n+1})|Y_{n+1} = Y(x_{n+1})\Big)dx$$

et ce qui est important, c'est que ce critère à en fait une forme analytique sous forme de fonctions de répartition gaussienne de dimension 2 (calculables avec le logiciel R), comme on peut le voir dans [**?**].

Lorsque nous savons qu'on peut rendre analytique une quantité de la forme $\mathbb{P}_n(Y(x) \leq a)$, on se dit tout de suite qu'on va pouvoir appliquer la méthode à la recherche du quantile puisque celui-ci est défini par ce type de probabilité. Nous choisissons alors le critère suivant, qui donne en moyenne l'erreur commise par notre estimateur (et qui est d'une forme semblable au critère précédent et qui sera donc certainement analytique) :

$$\Gamma_n = \int_{\mathbb{X}} |\mathbb{P}_n(Y(x) \geq \hat{q}^n_\alpha) - (1-\alpha)|dx$$

(où l'on rappelle que la loi sur $\mathbb{X}$ que nous considérons est la loi uniforme que $[0,1]^d$).

Ainsi, au début de l'étape $(n+1)$ il nous faudra choisir le x_{n+1} qui minimisera le critère de l'étape $(n+1)$ si l'on avait choisit ce point d'évaluation.

Nous cherchons alors une forme quasi-analytique du critère suivant :

$$\Gamma_{n+1} = \int_{\mathbb{X}} |\mathbb{P}_n\Big(Y(x) \geq \hat{q}_\alpha^{n+1} | Y(x_{n+1}) = Y_{n+1}\Big) - (1-\alpha)| dx$$

Mais surtout, nous cherchons, comme dans la méthode SUR, à exprimer ce critère calculé à l'étape $(n+1)$ comme un critère ne dépendant que de la tribu $\mathcal{F}_n$ et du point en lequel on l'évalue. De plus, l'objectif est le calcul de ces quantiles sans avoir à faire d'approximation Monte Carlo, ou moins en tout cas qu'avec la précédente méthode.

Nous devons donc d'une part faire disparaître tous les exposants $(n+1)$, d'autre part enlever toutes les dépendances en y_{n+1} (évaluation du point en lequel on calcule le critère) et enfin réduire en une forme analytique. Pour commencer, nous allons chercher à enlever les indices $(n+1)$ dans l'estimateur du quantile.

6.2 A la recherche d'une forme quasi-analytique pour ce critère

6.2.1 Mise à jour du quantile

La première chose à faire pour trouver cette forme quasi-analytique, c'est de mettre sous forme récursive l'estimateur du quantile. Lorsque nous considérons les deux estimateurs du quantile introduits dans la partie précédente, la solution n'est pas évidente et il semble que nous devrons forcément utiliser des étapes Monte Carlo. Ainsi, nous avons décidé de faire une hypothèse un peu abusive mais qui nous permettra d'attendre notre objectif (au moins une heuristique) : nous allons écrire cette formule de mise à jour en considèrant le quantile empirique. En effet, dans ce cas, tout se passe assez facilement, il suffit de faire un dessin et distinguer les cas.

Dans la suite, lorsque nous écrirons $y_{(i)}$, il s'agira de la $i^{ème}$ observation à l'étape n lorsque celles-ci sont classées par ordre croissant. Pour faire cette mise à jour, nous devons bien entendu différencier les cas selon où se situera notre nouvelle observation y_{n+1} par rapport aux n autres déjà ordonnées. Mais ils nous faudra aussi observer le comportement de la fonction $n \mapsto [n\alpha]]$.

1) Cas où $[(n+1)\alpha] = [n\alpha] + 1$:

Pour fixer les idées, regardons sur la figure 1 ce qu'il se passe lorsque $y_{n+1} \leq \hat{q}_\alpha^n = y_{[n\alpha]+1}$: $\hat{q}_\alpha^{n+1}$ sera alors la $[n\alpha]+2$-ème observation après avoir reclassé. Mais

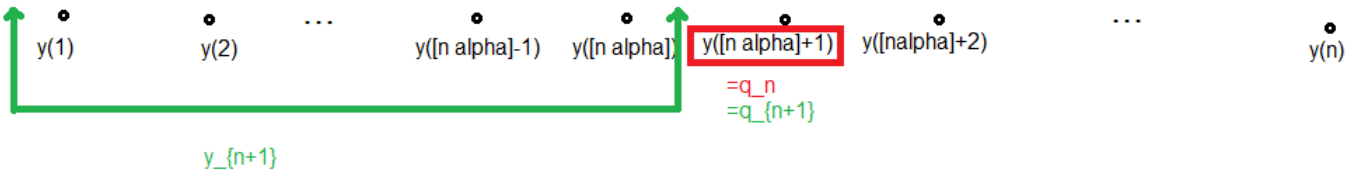

FIGURE 1 – Encadré en rouge, l'estimateur au pas n. L'accolade verte montre où peut se situer la (n+1)-ème observation pour que l'estimateur au pas (n+1) (vert) ne change pas.

comme y_{n+1} se situe avant la $[n\alpha]+1$-ème, notre estimateur va rester le même : $\hat{q}_\alpha^n = \hat{q}_\alpha^{n+1}$.

De la même manière on comprend bien que si $y_{n+1} \in]y_{([n\alpha]+1)}, y_{([n\alpha]+2)}[$ alors $\hat{q}_\alpha^{n+1} = y_{n+1}$ et si $y_{n+1} \geq y_{([n\alpha]+2)}$ alors $\hat{q}_\alpha^{n+1} = y_{([n\alpha]+2)}$. Il faut toutefois faire attention aux cas limites : si à l'étape n, l'estimateur du quantile est la plus grande observation y, alors les deux derniers cas se rassemblent en un seul cas : si la nouvelle observation y_{n+1} est la plus grande ($y_{n+1} \geq y_{(n)}$) alors l'estimateur du quantile à l'étape $(n+1)$ devient cette nouvelle observation elle-même : $\hat{q}_\alpha^{n+1} = y_{n+1}$.

2) Cas où $[(n+1)\alpha] = [n\alpha]$:

Pour traiter ce cas, il suffit de décaler les résultats de la partie précédente puisque à ce moment là, $\hat{q}_\alpha^{n+1}$ sera la $[n\alpha]+1$-ème observation après avoir reclassé. Nous trouvons donc que si $y_{n+1} \leq y_{([n\alpha])}$ alors $\hat{q}_\alpha^{n+1} = y_{([n\alpha])}$, si $y_{n+1} \in]y_{([n\alpha])}, y_{([n\alpha]+1)}[$ alors $\hat{q}_\alpha^{n+1} = y_{([n\alpha]+1)}$ et si $y_{n+1} \geq y_{([n\alpha]+1)}$ alors $\hat{q}_\alpha^{n+1} = \hat{q}_\alpha^n = y_{([n\alpha]+1)}$. De la même manière que précédemment, le cas limite où à l'étape n, $\hat{q}_\alpha^n = y_{(1)}$, les deux premiers cas se transforment en un unique : si $y_{n+1} \leq y_{(1)}$ alors $\hat{q}_\alpha^{n+1} = y_{n+1}$.

Nous avons alors tout ce qu'il faut pour écrire la formule de mise à jour du quantile empirique hors cas limites :

Si $[n\alpha]+1 = [(n+1)\alpha]$:

$$\hat{q}_\alpha^{n+1} = \hat{q}_\alpha^n 1_{y_{n+1} \leq \hat{q}_\alpha^n} + y_{n+1} 1_{y_{n+1} \in]y_{([n\alpha]+1)}, y_{([n\alpha]+2)}[} + y_{([n\alpha]+2)} 1_{y_{n+1} \geq y_{([n\alpha]+2)}}$$

Si $[n\alpha] = [(n+1)\alpha]$:

$$\hat{q}_\alpha^{n+1} = y_{([n\alpha])} 1_{y_{n+1} \leq y_{([n\alpha])})} + y_{n+1} 1_{y_{n+1} \in]y_{([n\alpha])}, y_{([n\alpha]+1)}[} + \hat{q}_\alpha^n 1_{y_{n+1} \geq y_{([n\alpha]+1)}}$$

Et lorsque l'estimateur est la plus grande observation et que $[n\alpha]+1 = [(n+1)\alpha]$:

$$\hat{q}_\alpha^{n+1} = \hat{q}_\alpha^n 1_{y_{n+1}\leq \hat{q}_\alpha^n} + y_{n+1} 1_{y_{n+1}\geq y_{(n)}}$$

Enfin lorsque l'estimateur est la plus petite observation et que $[n\alpha] = [(n+1)\alpha]$:

$$\hat{q}_\alpha^{n+1} = y_{n+1} 1_{y_{n+1}\leq y_{(1)}} + \hat{q}_\alpha^n 1_{y_{n+1}\geq y_{([n\alpha]+1)}}$$

6.2.2 Ecriture des probabilités conditionnelles comme des fonctions de répartitions bi-gaussiennes

Mise en avant des probabilités à simplifier :

Notre critère peut s'écrire aussi de la manière suivante :

$$\Gamma_{n+1} = \int_{\mathbb{X}} \left| \int_{\mathbb{R}} \mathbb{P}_n\Big(Y(x) \geq \hat{q}_\alpha^{n+1} | y_{n+1}) d\phi(y_{n+1}\Big) - (1-\alpha) \right| dx$$

où ϕ est la loi de $Y_n(x_{n+1})$ c'est-à-dire une loi $\mathcal{N}(m_n(x_{n+1}), s_n(x_{n+1}))$. Cette forme montre d'ailleurs mieux la dépendance en x_{n+1}, paramètre de la minimisation. On rappelle que nous avons un double objectif : on veut d'une part donner une forme simple au critère (quasi-analytique) et d'autre part enlever toute dépendance autre que celles en x, $\mathcal{F}_n$ et x_{n+1}.

Le but va alors être de simplifier l'expression :

$$P_n = \int_{\mathbb{R}} \mathbb{P}_n(Y(x) \geq \hat{q}_\alpha^{n+1} | y_{n+1}) d\phi(y_{n+1})$$

En remplaçant l'estimateur du quantile par sa formule récursive, nous obtenons, si $[(n+1)\alpha] = [n\alpha]+1$ (dans le cas standard) :

$$P_n = \int_{-\infty}^{y_{([n\alpha]+1)}} \mathbb{P}_n\Big(Y(x) \geq \hat{q}_\alpha^n | y_{n+1}\Big) d\phi(y_{n+1}) + \int_{y_{([n\alpha]+1)}}^{y_{([n\alpha]+2)}} \mathbb{P}_n\Big(Y(x) \geq y_{n+1} | y_{n+1}\Big) d\phi(y_{n+1})$$

$$+ \int_{y_{([n\alpha]+2)}}^{+\infty} \mathbb{P}_n\Big(Y(x) \geq y_{([n\alpha]+2)} | y_{n+1}\Big) d\phi(y_{n+1})$$

ce qui se réécrit :

$$P_n = \mathbb{P}_n\Big(Y(x) \geq \hat{q}_\alpha^n | Y(x_{n+1}) \leq y_{([n\alpha]+1)}\Big) + \mathbb{P}_n\Big(Y(x) \geq Y(x_{n+1} | Y(x_{n+1}) \in [y_{([n\alpha]+1)}, y_{([n\alpha]+2)}]\Big)$$

$$+ \mathbb{P}_n\Big(Y(x) \geq y_{([n\alpha]+2)} | Y(x_{n+1}) \geq y_{([n\alpha]+2)}\Big)$$

Et si $[(n+1)\alpha] = [n\alpha]$:

$$P_n = \int_{-\infty}^{y_{([n\alpha])}} \mathbb{P}_n\Big(Y(x) \geq y_{([n\alpha])}|y_{n+1}\Big)d\phi(y_{n+1}) + \int_{y_{([n\alpha])}}^{y_{([n\alpha])}} \mathbb{P}_n\Big(Y(x) \geq y_{n+1}|y_{n+1}\Big)d\phi(y_{n+1})$$
$$+ \int_{y_{([n\alpha]+1)}}^{+\infty} \mathbb{P}_n\Big(Y(x) \geq \hat{q}_\alpha^n|y_{n+1}\Big)d\phi(y_{n+1})$$

ce qui se réécrit :

$$P_n = \mathbb{P}_n\Big(Y(x) \geq y_{([n\alpha])}|Y(x_{n+1}) \leq y_{([n\alpha])}\Big) + \mathbb{P}_n\Big(Y(x) \geq Y(x_{n+1}|Y(x_{n+1}) \in [y_{([n\alpha])}, y_{([n\alpha]+1)}]\Big)$$
$$+ \mathbb{P}_n\Big(Y(x) \geq \hat{q}_\alpha^n|Y(x_{n+1}) \geq y_{([n\alpha]+1)}\Big)$$

Nous constatons alors que dans tous les cas, les probabilités à exprimer sont de la forme (aux signes près) $P_n^1 = \mathbb{P}_n\Big(Y(x) \leq a|Y(x_{n+1}) \leq b\Big)$ ou $P_n^2 = \mathbb{P}_n\Big(Y(x) \leq Y(x_{n+1})|Y(x_{n+1}\Big) \leq c)$. Nous allons donc essayer d'exprimer ces deux probabilités.

Simplification de P_n^1 :

On considère donc la probabilité suivante : $P_n^1 = \mathbb{P}_n\Big(Y(x) \leq a|Y(x_{n+1}) \leq b\Big)$.

Il est alors important de faire attention aux lois que nous considérons. En dehors du conditionnement ("connaissant $Y(x_{n+1})$"), $Y(x)$ suit une loi gaussienne de paramètres $m_n(x)$ et $s_n(x)$. Mais lorsque nous sommes sous le conditionnement, $Y(x)$ suit une loi gaussienne de paramètres $m_{n+1}(x)$ et $s_{n+1}(x)$ mais le premier paramètre (et seulement le premier !) dépend totalement de y_{n+1} ! Il va donc nous falloir utiliser les formules de mise à jour pour l'éspérance et la variance conditionnelle que nous avons vu en partie 2, pour enlever cette dépendance dans la partie qui sortira de l'intégrale.

Commençons par centrer et réduire la partie de gauche :

$$P_n^1 = \int_{-\infty}^{b} \mathbb{P}_n\Big(Y(x) \leq a\Big|y_{n+1})d\phi(y_{n+1}\Big)$$
$$P_n^1 = \int_{-\infty}^{b} \mathbb{P}_n\Big(\frac{Y(x) - m_{n+1}(x)}{s_{n+1}(x)} \leq \frac{a - m_{n+1}(x)}{s_{n+1}(x)}\Big|y_{n+1}\Big)d\phi(y_{n+1})$$

$m_{n+1}(x)$ dépendant de y_{n+1}, nous utilisons la formule de mise à jour de l'espérance : $m_{n+1}(x) = m_n(x) + \frac{c_n(x,x_{n+1})(Y(x_{n+1} - m_n(x_{n+1})}{s_n^2(x_{n+1})}$. Cela nous donne la simplification suivante :

$$\int_{-\infty}^{b} \mathbb{P}_n\Big(\frac{Y(x)-m_n(x)}{s_{n+1}(x)} - \frac{c_n(x,x_{n+1})(y_{n+1}-m_n(x_{n+1})}{s_n^2(x_{n+1})s_{n+1}(x)} \leq \frac{a-m_n(x)}{s_{n+1}(x)} - \frac{c_n(x,x_{n+1})(y_{n+1}-m_n(x_{n+1})}{s_n^2(x_{n+1})s_{n+1}(x)} \Big| y_{n+1}\Big) d\phi(y_{n+1})$$

D'où :

$$\int_{-\infty}^{b} \mathbb{P}_n\Big(\frac{Y(x)-m_n(x)}{s_{n+1}(x)} \leq \frac{a-m_n(x)}{s_{n+1}(x)} \Big| y_{n+1}\Big) d\phi(y_{n+1}) = \int_{-\infty}^{b} \mathbb{P}_n\Big(\frac{Y(x)-m_n(x)}{s_{n+1}(x)} \leq f_n^x(a) | y_{n+1}\Big) d\phi(y_{n+1})$$

et l'on gardera cette notation : $f_i^j(d) = \frac{d-m_i(j)}{s_i(j)}$.

Centrons et réduisons maintenant dans le conditionnement à présent :

$$P_n^1 = \mathbb{P}_n\Big(\frac{Y(x)-m_n(x)}{s_{n+1}(x)} \leq f_n^x(a) \Big| \frac{Y(x_{n+1})-m_n(x_{n+1})}{s_n(x_{n+1})} \leq f_n^{x_{n+1}}(b)\Big)$$

Ainsi :

$$P_n^1 = \int_{-\infty}^{f_n^{x_{n+1}}(b)} \mathbb{P}_n(Y \leq f_n^x(a)|X=u) \frac{\exp(-\frac{u^2}{2})}{\sqrt{2\pi}} du$$

où sachant la tribu $\mathcal{F}_n$, $X = \frac{Y(x_{n+1})-m_n(x_{n+1})}{s_n(x_{n+1})} \sim \mathcal{N}(0,1)$ et $Y = \frac{Y(x)-m_n(x)}{s_{n+1}(x)} \sim \mathcal{N}(0, \frac{s_n^2(x)}{s_{n+1}^2(x)})$.

Nous allons alors utiliser des résultats sur les vecteurs gaussiens. Nous savons en effet que (X,Y) est un vecteur gaussien centré de matrice de Covariance $K_{(X,Y)} = K_\beta = \begin{pmatrix} 1 & \beta \\ \beta & 1+\beta^2 \end{pmatrix}$, où l'on note $\beta = \frac{c_n(x,x_{n+1})}{s_{n+1}(x)s_n(x_{n+1})}$. En effet, X est centré réduit et Y est centré de variance $\mathrm{Var}(Y) = \frac{s_n^2(x)}{s_{n+1}^2(x)}$. Mais nous pouvons simplifier cette variance à l'aide la deuxième formule de mise à jour, celle sur la variance : $s_{n+1}^2(x) = s_n^2(x) - \frac{c_n^2(x,x_{n+1}}{s_n^2(x_{n+1}}$). Ainsi, $\mathrm{Var}(Y) = 1+\beta^2$ et $Cov(X,Y) = \beta$. Ainsi, la loi de Y sachant $X = x$ est une loi normale de paramètres $m_X = \frac{Cov(X,Y)x}{var(X)} = \beta x$ et $\sigma^2 = \mathrm{Var}(Y) - \frac{\mathrm{Cov}(X,Y)^2}{Var(X)^2} = 1$. Finalement : $\mathcal{L}(Y|X=x) = \mathcal{N}(\beta x, 1)$. Cela nous permet alors de terminer la simplification de notre probabilité :

$$P_n^1 = \int_{-\infty}^{f_n^{x_{n+1}}(b)} \int_{-\infty}^{f_n^x(a)} \exp(-\frac{1}{2}((y-\beta u)^2 + u^2)) \frac{dydu}{2\pi}$$

Et par Fubini :

$$P_n^1 = \int_{]-\infty, f_n^{x_{n+1}}(b)] \times]-\infty, f_n^x(a)]} \exp(-\frac{1}{2}(u,y)K_\beta^{-1}(u,y)^T) \frac{dudy)}{2\pi} = \Phi_{K_\beta}\Big(f_n^{x_{n+1}}(b), f_n^x(a)\Big)$$

Où Φ_{K_β} est la fonction de répartition d'un vecteur gaussien centré de matrice de covariance K_β avec β ne dépendant que de x, $\mathcal{F}_n$ et x_{n+1}.

Nous avons donc réussi à simplifier notre première probabilité. La forme n'est pas encore totalement analytique mais il existe des packages R qui vont nous permettre de très bien approximer ces expressions.

Simplification de P_n^2

La deuxième sorte de probabilité que nous avons à simplifier est de la forme suivante :

$$P_n^2 = \mathbb{P}_n\Big(Y(x) \leq Y(x_{n+1})\Big|Y(x_{n+1}) \leq b\Big)$$

L'idée est la suivante : nous allons essayer d'enlever la dépendance en l'évaluation de Y en x_{n+1} dans la partie conditionnement pour se ramener à une probabilité de la forme de la précédente. Pour cela, nous allons considérer d'une part la même variable aléatoire :

$$X = \frac{Y(x_{n+1}) - m_n(x_{n+1})}{s_n(x_{n+1})}$$

suivant une loi normale centrée réduite et d'autre part, la variable aléatoire :

$$Y' = \frac{Y(x) - m_{n+1}(x)}{s_{n+1}(x)} - \frac{Y(x_{n+1}) - m_n(x_{n+1})}{s_{n+1}(x)}$$

qui est centrée et de variance :

$$\mathrm{Var}(Y') = \frac{s_n^2(x)}{s_{n+1}^2(x)} + \frac{s_n^2(x_{n+1})}{s_{n+1}^2(x)} - 2\frac{c_n(x, x_{n+1})}{s_{n+1}^2(x)}.$$

Ainsi :

$$\mathrm{Cov}(X, Y') = \frac{c_n(x, x_{n+1}) - s_n^2(x_{n+1})}{s_{n+1}(x)s_n(x_{n+1})}$$

que l'on appelle ν. Nous avons alors avec cette notation $\mathrm{Var}(Y') = 1 + \nu^2$ et

$$K_{(X,Y')} = K_\nu = \begin{pmatrix} 1 & \nu \\ \nu & 1+\nu^2 \end{pmatrix}$$

.

Et alors en utilisant la formule de mise à jour de l'espérance exactement comme pour P_n^1 :

$$P_n^2 = \mathbb{P}_n\Big(Y' \leq z_n(x)\Big|X \leq f_n^{x_{n+1}}(b)\Big)$$

avec $z_n(x) = \frac{m_n(x_{n+1}) - m_n(x)}{s_{n+1}(x)}$ (qui ne dépend plus de $Y(x_{n+1})$).

Nous sommes finalement ramené à simplifier une probabilité de la forme de P_n^1 avec X et Y'. En suivant exactement le même chemin que dans la partie précédente, nous trouvons alors que :

$$P_n^2 = \Phi_{K_\nu}\Big(f_n^{x_{n+1}}, z_n(x)\Big)$$

où à nouveau, ν ne dépend que des bons paramètres.

Et si les signes changent ?

On remarque assez facilement que une fois que nous avons P_n^1 et P_n^2, nous pouvons calculer toutes les probabilité dérivées, en changeant les signes : si $P_n\Big(Y \leq a|X \leq b\Big) = \Phi_{K_\beta}(b', a')$ alors $P_n(Y \geq a|X \leq b) = \Phi_{K_{-\beta}}(b', -a')$, $P_n(Y \leq a|X \geq b) = \Phi_{K_{-\beta}}(-b', a')$ et $P_n(Y \geq a|X \geq b) = \Phi_{K_\beta}(-b', -a')$;

Nous pouvons donc calculer toutes les probabilités nécessaires pour simplifier notre critère.

6.2.3 Le critère simplifié

Finalement, nous trouvons que si nous ne sommes pas dans un cas limite et que $[(n+1)\alpha] = [n\alpha] + 1$ alors :

$$\Gamma_{n+1} = \int_{\mathbb{X}} \Big|\Phi_{K_{-\beta}}\Big(f_n^{x_{n+1}}(y_{([n\alpha]+1)}), f_n^x(-\hat{q}_\alpha^n)\Big) + \Phi_{K_{-nu}}(f_n^{x_{n+1}}\Big(y_{([n\alpha]+2)}), -z_n(x)\Big)$$

$$-\Phi_{K_{-\nu}}\Big(f_n^{x_{n+1}}(y_{([n\alpha]+1)}), -z_n(x)\Big)+\Phi_{K_\beta}\Big(-f_n^{x_{n+1}}(y_{([n\alpha]+2)}), -f_n^x(y_{([n\alpha]+2)}\Big)-(1-\alpha)\Big|dx$$

et que si $[(n+1)\alpha] = [n\alpha]$:

$$\Gamma_{n+1} = \int_{\mathbb{X}} \Big|\Phi_{K_{-\beta}}\Big(f_n^{x_{n+1}}(y_{([n\alpha])}), f_n^x(-y_{([n\alpha])})\Big) + \Phi_{K_{-nu}}\Big(f_n^{x_{n+1}}(y_{([n\alpha]+)}), -z_n(x)\Big)$$

$$-\Phi_{K_{-\nu}}\Big(f_n^{x_{n+1}}(y_{([n\alpha])}), -z_n(x)\Big) + \Phi_{K_\beta}\Big(- f_n^{x_{n+1}}(y_{([n\alpha]+1)}), -f_n^x(\hat{q}_\alpha^n)\Big) - (1-\alpha)\Big|dx$$

et de la même manière, sans détailler les calculs, si nous sommes dans les cas limites vus précédemment, si $[(n+1)\alpha] = [n\alpha] + 1$ et que l' estimateur était la plus grande observation :

$$\Gamma_{n+1} = \int_{\mathbb{X}} \Big|\Phi_{K_{-\beta}}\Big(f_n^{x_{n+1}}(y_{([n\alpha]+1)}), f_n^x(-\hat{q}_\alpha^n)\Big)+\Phi_{K_{nu}}\Big(f_n^{x_{n+1}}(-y_{([n\alpha]+1)}), -z_n(x)\Big)-(1-\alpha)\Big|dx$$

et si $[(n+1)\alpha] = [n\alpha]$ et que l'éstimateur était la plus petite observation :

$$\Gamma_{n+1} = \int_{\mathbb{X}} \left| \Phi_{K_{-\nu}}\left(f_n^{x_{n+1}}(y_{([n\alpha]+1)}), -z_n(x)\right) + \Phi_{K_\beta}\left(-f_n^{x_{n+1}}(y_{([n\alpha]+1)}), -f_n^x(\hat{q}_\alpha^n)\right) - (1-\alpha) \right| dx$$

Il reste maintenant à mettre ceci en algorithme. Nous avons à disposition un package R qui nous permet d'estimer les fonctions de répartition bi-gaussiennes.

6.3 L'algorithme de la nouvelle méthode

Pour programmer la méthode, j'avais donc besoin à chaque étape n, de calculer les paramètres de la loi normale venant des formules de krigeage. Dans un premier temps, j'ai créé une fonction nommée "loiCond" qui prend comme paramètres x (matrice donnant les points évalués de taille d×n), y (vecteur des évaluations de taille n) et xnew et renvoie une liste contenant le vecteur des moyennes conditionnelles de xnew et la matrice de covariance conditionnelle de xnew. Ainsi, pour choisir le nouveau point, lorsque j'avais besoin des paramètres de krigeage, j'appelais cette fonction. J'ai aussi bien sûr créé une fonction "est" qui prenait en entrée un vecteur y d'évaluations et sortait l'estimateur empirique. J'ai alors pu écrire le programme suivant :

1. On génère une grille *xsuiv* de 500 points sur $[0,1]^d$ qui sera la grille sur laquelle on choisira les nouveaux points à évaluer.
2. On initialise notre vecteur x des points évalués en choisissant les n_0 premiers points de *xsuiv* . On évalue ses points et on met les évaluations dans un vecteur y.
3. On génère une grille *xgrid* de points de $[0,1]^d$ qui nous permettra d'estimer l'intégrale sur $\mathbb{X}$.
4. Pour n allant de n_0 à n_{budget} :
   a. On passe dans la fonction "nextPoint" qui prend en entrées x, y, *xsuiv* et *xgrid* courants, et renvoie le nouveau point à évaluer.
   b. On ajoute le point à x, on l'évalue, on ajoute le résultat à y et on calcule l'estimateur courant.
   c. On enlève le point choisi de la grille *xsuiv* .
5. On renvoie est(y).

où la fonction principale "nextPoint" est de la forme suivante :

Entrées : *xgrid* , x, y, *xsuiv* courants.

I. Boucle sur les *xsuiv* (dans le but de remplir un vecteur *intégrale* qui nous donne en coordonnée i, l'approximation de l'intégrale si x_{n+1} est le *xsuiv* courant).

A. Passage dans loiCond pour récupérer les coefficients utiles au calcul des fonctions de répartition, ne dépendant que de *xsuiv* .

B. Boucle sur les *xgrid* (dans le but de remplir un vecteur *contrib* qui aura en coordonnée k la contribution de l'intégrale donnée par le *xgrid* courant en considérant que x_{n+1} est le *xsuiv* courant).

   a. Appel de la fonction "loiCond" pour récupérer les paramètres nécessaires aux calculs des fonctions de répartitions bi-gaussiennes.

   b. Calculs des fonctions de répartition bi-gaussiennes avec le package bivnorm de R (en distinguant les deux cas sur la partie entière de $n \times \alpha$).

   c. Calcul de la contribution courante (en distinguant si l'estimateur courant est la plus petite, la plus grande ou une autre observation).

C. Calcul de l'approximation de l'intégrale courante grâce au vecteur des contributions.

II. Recherche du *xsuiv* qui minimise le vecteur *intégrale*.

Sortie : Le nouveau point à évaluer.

Nous constatons d'abord que la complexité de cet algorithme est meilleure que celle du précédent puisque nous n'avons plus que deux boucles imbriquées. Nous verrons dans la partie suivante comme cela améliore les temps de calculs. Cependant, cet algorithme nous oblige encore à passer par le programme loiCond qui, en utilisant les formules brutes de krigeage, procède à l'inversion d'une matrice $n \times n$, ce qui est assez coûteux. Mais comme à chaque étape, nous ne voulons trouver que **le** point suivant à évaluer, nous pouvons éviter cette inversion en utilisant à nouveau les formules de mise à jour à un pas que nous avons vu précédemment.

6.4 L'algorithme de la nouvelle méthode avec formules de mise à jour

Le programme est un peu plus compliqué dans le mesure où il va falloir à chaque étape n, stocker tous les coefficients de krigegage calculés, pour les utiliser à l'étape (n+1). Mais le gain de complexité sera vraiment important. Nous devons bien entendu tout de même passer par le programme loiCond au départ, pour initialiser les formules de mise à jour, et nous aurons toujours besoin de la fonction *est*. Voici comment j'ai construit la boucle principale :

1. On tire une grille *xsuiv* de 500 points sur $[0,1]^d$ qui sera la grille sur laquelle on choisira les nouveaux points à évaluer.
2. On initialise notre vecteur x des points évalués en choisissant les $(n_0 - 1)$ premiers points de *xsuiv* (pour initialiser plus facilement, on va faire comme

si le premier point sélectionné était le n_0-ième de *xsuiv* . On évalue ses points et on met les évaluations dans un vecteur y.

3. On tire une grille *xgrid* de points de $[0,1]^d$ qui nous permettra d'estimer l'intégrale sur $\mathbb{X}$.
4. On appelle la fonction *loicond* pour récupérer tous les coefficients dont nous aurons besoin pour la première étape (en faisant comme si le point selectionné à cette étape-ci était $xsuiv[n_0,]$).
5. On évalue $xsuiv[n_0,]$ et on met à jour x et y. On calcule *est(y)* courant.
6. Pour n allant de n_0 à n_{budget} :

    a. On passe dans la fonction "nextPoint" qui a pour entrées x, y, *xsuiv* , *xgrid* courants, et toutes les variables préalablement stockées et nécessaires à l'étape en cours et qui renvoie une liste contenant le nouveau point à évaluer et tous les variables stockées au cours de l'étape.

    b. On ajoute le point à x, on l'évalue, on ajoute le résultat à y et on calcule l'estimateur courant.

    c. On récupère les informations stockées nécessaires (pour les donner en paramètre de nextPoint à la prochaine étape)

    c. On enlève le point choisi de le grille *xsuiv* .

5. On renvoie est(y).

où la fonction principale "nextPoint" est de la forme suivante :

Entrées : *xgrid* , x, y, *xsuiv* courants et toutes les variables stockées à l'étape précédente nécessaires à cette étape.

I. Récupération des variables de stock entrées en paramètres dans des variables "boucle" et réinitialisation des variables de stock pour garder les coefficients de l'étape courante nécessaires à la prochaine.

II. Boucle sur les *xsuiv* (dans le but de remplir un vecteur *intégrale* qui nous donne en coordonnée i, l'approximation de l'intégrale si x_{n+1} est le *xsuiv* courant).

    A. Calcul par formules de mise à jour des coefficients utiles au calcul des fonctions de répartitions, ne dépendant que de *xsuiv* . Stockage.

    B. Boucle sur les *xgrid* (dans le but de remplir un vecteur *contrib* qui aura en coordonnée k la contribution de l'intégrale donnée par le *xgrid* courant en considérant que x_{n+1} est le *xsuiv* courant).

        a. Calcul par formules de mise à jour des coefficients utiles au calcul des fonctions de répartition bi-gaussiennes. Stockage.

b. Calculs des fonctions de répartition bi-gaussiennes avec le package *bivnorm* de R (en distinguant les deux cas sur la partie entière de $n \times \alpha$).

c. Calcul de la contribution courante (en distinguant si l'estimateur courant est la plus petite, la plus grande ou une autre observation).

C. Calcul de l'approximation de l'intégrale courante grâce au vecteur des contributions.

II. Recherche du *xsuiv* qui minimise le vecteur intégrale.

Sortie : Une liste contenant le nouveau point à évaluer et les variables stockées.

Ces programmes sont disponibles en annexe.

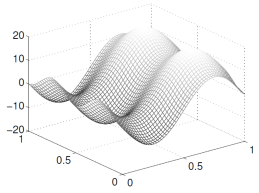

FIGURE 2 – Fonction de Ishigami

7 Résultats

Nous avons donc construit une nouvelle méthode qui semble plus rapide que la précédente. Nous voulons maintenant comparer les résultats de ces deux méthodes. Pour commencer, nous allons voir ce qu'il se passe en dimension 2, puisque c'est dans cette dimension que les résultats de l'équipe ont été publiés dans [**?**]. Ensuite nous verrons comment on pourrait essayer d'implémenter, en dimension supérieure, la nouvelle méthode.

7.1 Comparaison des résultats en dimension 2

Pour étudier les méthodes en dimension 2, nous allons comparer les résultats de la méthode SUR rapportés dans l'article [**?**] aux résultats de la nouvelle méthode. Dans cet article, on considère une grille $\mathcal{X}$ de 500 points et les deux Monte Carlo se font chacun avec 10 itérations. Ainsi, pour essayer d'avoir une équivalence de précision avec ma méthode, j'ai pris une taille de grille de 100 points pour estimer mon intégrale sur $\mathbb{X}$. Pour faire ces grilles, nous utilisons la fonction improvedLHS du logiciel R. Nous utilisons un noyau gaussien de paramètre $l = 0.15$: $k(x, y) = \exp(\frac{-||x-y||^2}{2l^2})$. Suivant les exemples, nous allons prendre $n0$=1 ou 20 et n_{budget}= 40, 80 ou 100. Finalement nous cherchons à estimer le quantile à 95 %.

7.1.1 Etude de la fonction de Ishigami

Nous considérons dans un premier temps la fonction de Ishigami (voir figure 2) :

$$f(x, y) = \sin(\pi(2x - 1)) + 7\sin(\pi(2y - 1))^2 + 0.1\pi^4 \sin(\pi(2x - 1))$$

Dans un premier temps, nous regardons, pour $n_0 = 10$ et $n_{\text{budget}} = 40$ (c'est assez pour obtenir une bonne approximation), la répartition des points générés par l'algorithme (figures 3 et 4).

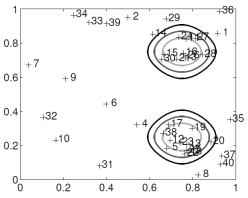

FIGURE 3 – Points générés pour la fonction de Ishigami avec la méthode SUR

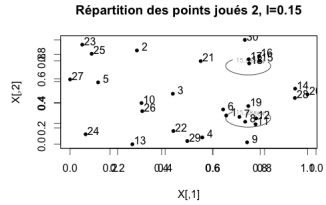

FIGURE 4 – Points générés pour la fonction de Ishigami avec la nouvelle méthode

Nous constatons que les deux méthodes donnent de bons résultats : on génère beaucoup de points proches de la ligne de niveau correspondant au quantile et quelques uns ailleurs pour ne pas rater d'endroit importants.

Nous allons ensuite regarder l'erreur relative du quantile courant au fur et à mesure des itérations des algorithmes. Dans l'article de Télécom, cette erreur relative est calculée par rapport au quantile estimé avec les 500 points de la grille (noté q_{500}). Pour comparer les deux méthodes, j'ai donc fait les mêmes expériences, mais j'ai aussi regardé l'erreur empirique par rapport au vrai quantile (estimé par un gros Monte Carlo de 10^5), ce qui me semblait plus intéressant (figures 5, 6 et 7).

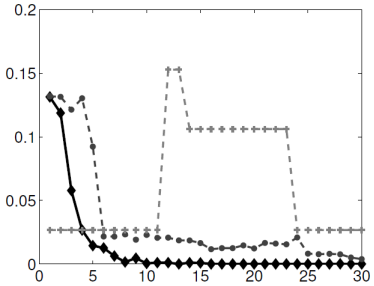

FIGURE 5 – Erreur relative par rapport à q_{500} avec la méthode SUR

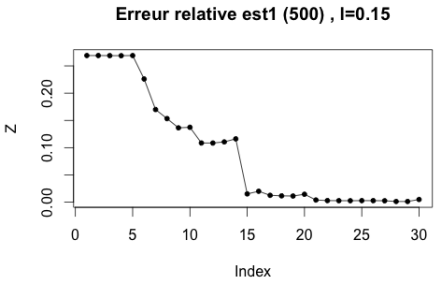

FIGURE 6 – Erreur relative par rapport à q_{500} : nouvelle méthode avec estimateur 1

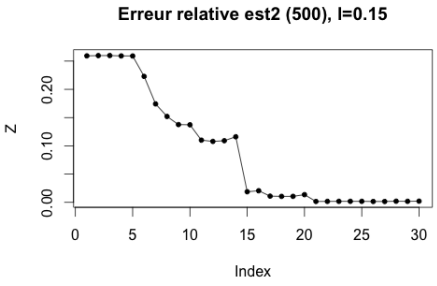

FIGURE 7 – Erreur relative par rapport à q_{500} : nouvelle méthode avec estimateur 2

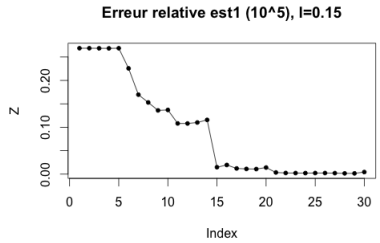

FIGURE 8 – Erreur relative par rapport au vrai quantile pour nouvelle méthode avec estimateur 1

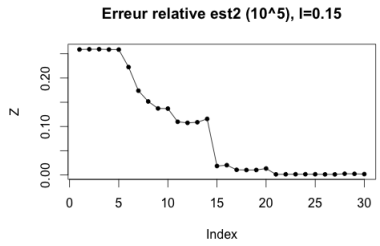

FIGURE 9 – Erreur relative par rapport au vrai quantile pour nouvelle méthode avec estimateur 2

Nous constatons que les deux méthodes fonctionnent très bien : au bout de 30 itérations des programmes, nous sommes à 1% d'erreur relative (et on passe sous les 5% au bout de 10 itérations pour SUR et 20 itérations pour l'autre méthode). Nous constatons de plus que pour la nouvelle méthode, les deux estimateurs sont équivalents dans ce cas.

Voyons l'erreur relative par rapport au vrai quantile avec la nouvelle méthode (Figure 8 et 9).

Les résultats sont sensiblement les mêmes.

7.1.2 La fonction de Gramacy

Cette fonction est celle qui fonctionne le moins bien avec les deux méthodes. Nous l'avons testée en prenant 20 points d'initialisation. Les résultats de la méthode SUR étaient donnés avec 50 itérations de l'algorithme, je suis allée jusqu'à 80 pour augmenter la précision, mais ce n'est pas beaucoup mieux.

La fonction de Gramacy est la suivante (voir figure 10) :

$$f(v, w) = (8v - 2)\exp(-(8v - 2)^2 - (8w - 2)^2)$$

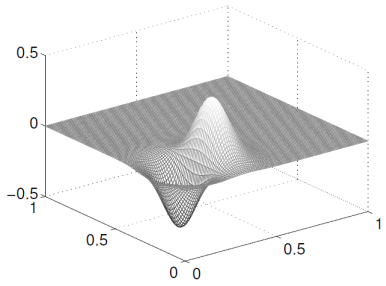

FIGURE 10 – Fonction de Gramacy

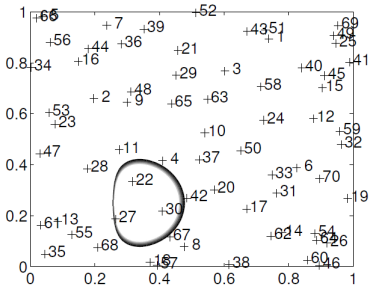

FIGURE 11 – Répartition des points avec la méthode SUR

On comprend que notre difficulté vient du fait que cette fonction est quasiment plate partout sauf à un endroit où elle a une grosse *bosse*. Observons la répartition des points générés par l'algorithme (figures 11 et 12).

Nous remarquons déjà que les méthodes ne fonctionnent pas très bien parce que les points ne sont pas concentrés dans les zones d'intérêt. Voyons comment cela se traduit sur les erreurs relatives (figures 13, 14, 15, 16 et 17).

Nous constatons que pour les deux méthodes, nous arrivons difficilement à 20% d'erreur relative par rapport à q_{500}, ce qui est très mauvais. En revanche, lorsque nous comparons avec le vrai quantile, le premier estimateur nous permet de passer sous la barre des 5% en 65 itérations. C'est mieux. On ne notera pas de différences significatives entre les deux estimateurs pour la nouvelle méthode.

7.1.3 Fonction de Sobol

Pour terminer, j'ai testé la nouvelle méthode sur la fonction de Sobol, ainsi définie : $f(x, y) = |4x - 2| \times |4y - 2|$. Nous ne disposons ici que des résultats de la nouvelle méthode. Observons la très bonne répartition des points ainsi que l'erreur relative qui en 30 itérations passe sous les 2% pour q_{500} et moins de 1% pour le

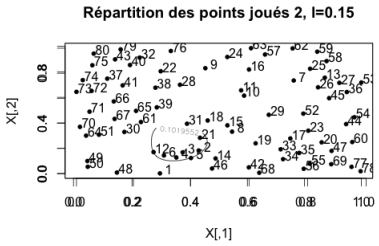

FIGURE 12 – Répartition des points avec la nouvelle méthode

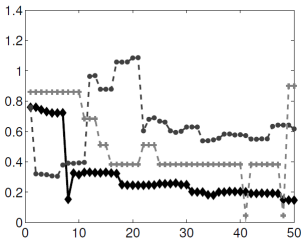

FIGURE 13 – Erreur relative q_{500} par la méthode SUR, en noir : SUR, en gris foncé : GPExplor, en gris clair : Random

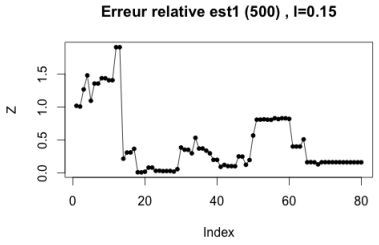

FIGURE 14 – Erreur relative q_{500} par la nouvelle méthode avec estimateur 1

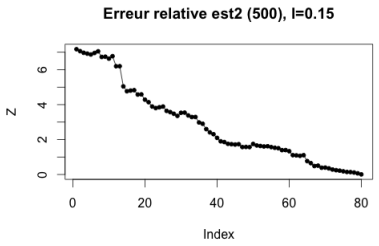

FIGURE 15 – Erreur relative q_{500} par la nouvelle méthode avec estimateur 2

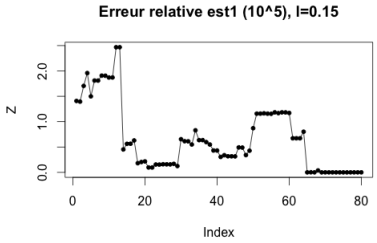

FIGURE 16 – Erreur relative par rapport au vrai quantile par la nouvelle méthode avec estimateur 1

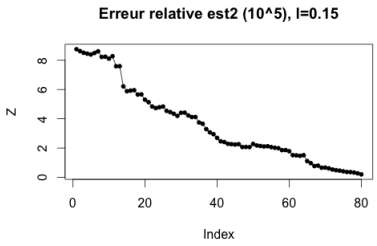

FIGURE 17 – Erreur relative par rapport au vrai quantile par la nouvelle méthode avec estimateur 2

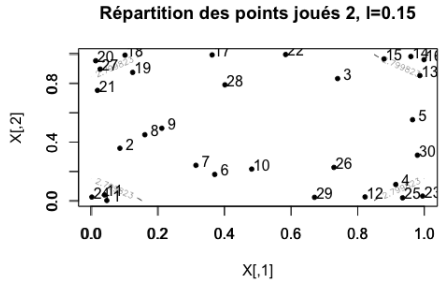

FIGURE 18 – Repartition des points générés par la nouvelle méthode

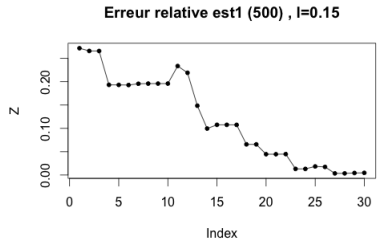

FIGURE 19 – Erreur relative par rapport à q_{500} pour l'estimateur 1

vrai quantile. Les résultats (figures 18, 19, 20, 21 et 22) pour cette fonction sont excellents.

7.1.4 Discussion sur les temps de calcul

Nous avons donc vu que les deux fonctions donnaient de très bons résultats en dimension 2, mise à part avec la fonction de Gramacy. Nous avons donc trouvé une méthode qui avait la même précision que la méthode SUR et notre objectif était de réduire le temps de calcul. Les deux boucles en moins se font nettement ressentir

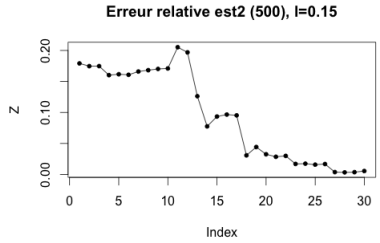

FIGURE 20 – Erreur relative par rapport à q_{500} pour l'estimateur 2

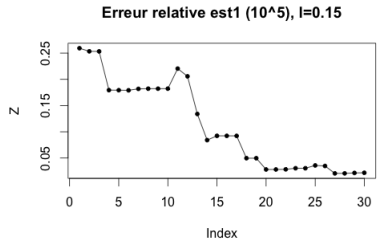

FIGURE 21 – Erreur relative par rapport au vrai quantile pour l'estimateur 1

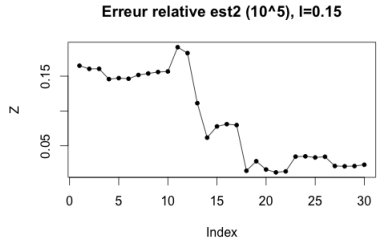

FIGURE 22 – Erreur relative par rapport au vrai quantile pour l'estimateur 2

dans les temps de calcul. En effet, l'expérience de la fonction de Ishigami prend 4h00 avec la méthode SUR et seulement 40 minutes avec la nouvelle méthode. Pour la fonction de Gramacy, on passe de 5h30 à 1h30 (pour 20 + 50 points) ou 2h15 (pour 20+80 points). Cette grosse réduction de temps de calcul nous donne un espoir quant à la possibilité de faire tourner l'algorithme en dimension 5, chose qui n'était pas possible avec la méthode SUR.

7.2 Résultats en dimension 5

N'ayant pas d'ordinateur assez performant pour lancer le programme, je me suis contentée d'évaluer le temps de calcul pour faire la même chose avec la fonction de Ishigami qu'en dimension 2 mais en dimension 5. Mon ordinateur a tourné 8h avant de me renvoyer des courbes qui atteignent les 50% d'erreur relative. Cette erreur n'est pas vraiment à prendre en compte puisque l'algorithme a tourné sur une grille de 500 points ce qui n'a pas vraiment de sens en dimension 5. Mais le temps obtenu laisse à penser qu'il serait intéressant de lancer l'algorithme avec une plus grosse grille, sur un ordinateur plus performant, d'autant plus que l'entreprise Orange autorise un temps de calcul d'une journée pour trouver à chaque étape le nouveau point.

Pour information, voici les courbes de la dimension 5, peu intéressantes (on

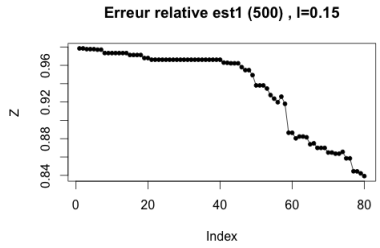

FIGURE 23 – Erreur relative par rapport à q_{500} pour estimateur 1

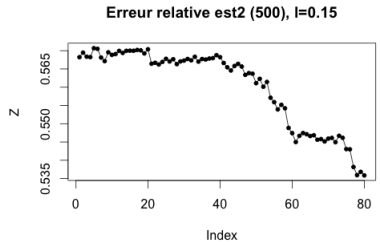

FIGURE 24 – Erreur relative par rapport à q_{500} pour estimateur 2

remarquera que dans ce cas-là, c'est l'estimateur 2 qui permet une plus grande précision).

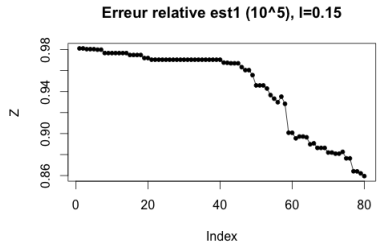

FIGURE 25 – Erreur relative par rapport au vrai estimateur pour estimateur 1

FIGURE 26 – Erreur relative par rapport au vrai estimateur pour estimateur 2

8 Conclusion

Mon stage s'inscrivait dans un projet de Orange Labs, ayant pour but d'estimer le quantile de la distribution de la quantité d'ondes électromagnétiques qu'absorde un foetus dans le ventre de sa mère lorsque cette dernière utilise un téléphone portable. Chaque expérience pour mesurer un tel taux étant très coûteuse, nous recherchions une méthode pour avoir une bonne estimation de ce quantile, en évaluant au plus une centaine de fois la fonction (en dimension 5).

Mon stage a globalement consisté à trouver une stratégie de choix de points à évaluer tout aussi efficace que la méthode SUR mais qui tournerait beaucoup plus vite. Pour cela, nous avons dû faire une grosse hypothèse qui n'est pas totalement justifiable, mais l'algorithme obtenu répond à la problématique.

Il resterait maintenant à essayer d'affiner cet algorithme en remplaçant la formule de mise à jour du quantile empirique par une formule de mise à jour avec un meilleur estimateur du quantile. De plus, nous pourrions réfléchir à diminuer encore la complexité de l'algorithme, en utilisant une méthode plus intelligente que le calcul sur toute une grille pour obtenir un minimum (par exemple un algorithme stochastique ou une méthode par dichotomie).

9 Remerciements

Je souhaite remercier mes deux maîtres de stage Fabrice Gamboa et Aurélien Garivier qui m'ont été d'une grande aide et d'un grand soutient pendant ce stage. Durant ces quatre mois, j'ai appris avec Aurélien à utiliser le logiciel R et je le remercie vivement pour tous les conseils qu'il a pu me donner et la patience dont il a fait preuve pour m'aider à corriger mes erreurs de programmation.

Enfin, je remercie Victor Picheny pour le séminaire vraiment très intéressant qu'il nous a présenté à l'IMT et qui m'a totalement inspiré la nouvelle méthode.

10 Bibliographie

Références

[1] Julien Bect, David Ginsbourger, Ling Li, Victor Picheny, and Emmanuel Vazquez. Sequential design of computer experiements for the estimation of a probability of failure. *Springer Science*, Avril 2011.

[2] Clément Chevalier. Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set. *Technometrics*.

[3] Clément Chevalier and David Ginsbourger. Corrected kriging update formulae for batch-sequential data assimilation. *ArXiv :1203.6452v1*, Mars 2012.

[4] Jean-Paul Chilés and Pierre Delfiner. *Geostatistics Modeling Spatial Uncertainty*. John Wiley, 1999.

[5] A. W. Van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

[6] Yves Gratton. Le krigeage : la methode optimale d'interpolation spatiale.

[7] Marjorie Jala, Celine Levy-Leduc, Eric Moulines, Emmanuelle Conil, and Joe Wiart. Sequential design of computer experiments for parameter estimation with application to numerical dosimetry. Août 2012.

[8] D. Marcotte. Krigegage-polycopié de cours pour l'école polytechnique. 2013.

[9] Victor Picheny. Séminaire institut mathématiques toulouse. 2013.

11 Annexes

Voici les codes que j'ai écrits durant mon stage.

11.1 Le programme de Télécom

```
 library(MASS)
library(lhs)
options(mc.cores= 8)
l <- 0.15;

mynorm <- function(x){
  sqrt(sum(x^2));
}

kern <- function(x1, x2){
  exp(-mynorm(x1-x2)^2/(2*l^2));
}

Kvec <- function(x1, x2){
  m <- dim(x1)[1];
  n <- dim(x2)[1];
  K <- matrix(0, nrow=m, ncol=n);
  for (i in 1:m){
    for (j in 1:n){
      K[i,j] <- kern(x1[i,], x2[j,]);
    }
  }
  K;
}

loiCond <- function(x, y, xnew){
  # in: features x = , values y, new features xnew
  res <- list();
  K <- Kvec(x, x);
  k <- Kvec(x, xnew);
  knew <- Kvec(xnew, xnew);
  res$esp <-  t(k) %*% solve(K, y);
  res$cov <- knew - t(k) %*% solve(K, k);
  res;
}
```

```
est <- function(y){
  alpha <- 0.95;
  z <- sort(y);
  z[floor(length(y)*alpha)+1];
}

nextPointMoulines <- function(x,y){
  # in: x,y = previous observations
  # out: next point to explore
  res<-list();
  d=dim(x)[2];
  N <- 10; # number of paths
  N2 <- 10; # number of replications for MC estimation of estimate variance.
  n<-dim({ \it xgrid })[1];

  loi <- loiCond(x, y, { \it xgrid });
  ygrid <- mvrnorm(N, loi$esp, loi$cov);
  varest <- array(n);
  for (k in 1:n){

    xn <- { \it xgrid }[k,];
    x2 <- rbind(x, xn);
    varest2 <- array(N);

     for (i in 1:N){
      y2 <- matrix(c(y, ygrid[i, k]));

      loi2 <- loiCond(x2, y2, { \it xgrid });
      est2 <- array(0, N2);
        for (j in 1:N2){
        ygrid2 <- mvrnorm(1, loi2$esp, loi2$cov);
        est2[j] <- est(ygrid2)
      }
      varest2[i] <- var(est2); # trŕs trŕs bizarre 1/2
    }
    varest[k] <- mean(varest2); # trŕs trŕs bizarre 2/2
  }
  kopt <- which.min(varest);
  res$indice<-kopt;
```

```
  res$point<-{ \it xgrid }[kopt,];
  res;
}


mcnextPointMoulines <- function(x,y){
  # in: x,y = previous observations
  # out: next point to explore
  N <- 50; # number of paths
  N2 <- 50; # number of replications for MC estimation of estimate variance
  { \it xgrid } <- seq(0.0, 1, 0.041);
  { \it xgrid } <- { \it xgrid }[sapply({ \it xgrid }, function(xn) all(abs(x-xn)>1
  n <- length({ \it xgrid });
  loi <- loiCond(x, y, matrix({ \it xgrid }));

  ygrid <- mvrnorm(N, loi$esp, loi$cov);

  varest <- mclapply(1:n, function(k){
    xn <- { \it xgrid }[k];
    x2 <- rbind(x, xn);
    varest2 <- array(N);
    for (i in 1:N){
      y2 <- c(y, ygrid[i, k]);
      loi2 <- loiCond(x2, y2, { \it xgrid });
      est2 <- array(0, N2);
      for (j in 1:N2){
        ygrid2 <- mvrnorm(1, loi2$esp, loi2$cov);
        est2[j] <- est({ \it xgrid }, ygrid2)
      }
      varest2[i] <- var(est2);     }
    mean(varest2);
  });
  print(varest);
  kopt <- which.min(varest); print(kopt);
  { \it xgrid }[kopt];
}


nextPointGPExplor<- function(x,y){
  d=dim(x)[1]
```

```
  { \it xgrid } <- t(matrix(runif(d, min=0, max=1)));
  for (i in 1:99) {
    { \it xgrid }<-rbind({ \it xgrid }, t(matrix(runif(d, min=0, max=1))))
  }
  #{ \it xgrid } <- { \it xgrid }[sapply({ \it xgrid }, function(xn) all(abs(x-xn)>
  n<-dim({ \it xgrid })[1];
  loi<-loiCond(x,y,{ \it xgrid });
  sigma<-diag(loi$cov);
  k<-which.max(sigma);
  { \it xgrid }[k,];
}



#nextPoint <- nextPointMoulines;
nextPoint <- nextPointMoulines;
# nextPoint <- nextPoint0;
nbudget<-50;
print(proc.time());
f <- function(x) mynorm(x) + sin(3*mynorm(x)*pi);
m=2;
x <- runif(5, min=0, max=1);
for (j in 1:(m-1)){
  x<-rbind(x, runif(5, min=0, max=1));
}

y<-f(x[1,]);
for (i in 2:m){
  y<-c(y, f(x[i,]));
}

d<-dim(x)[2];
n<-dim(x)[1];
ngrid<-100;
{ \it xgrid }<-randomLHS(ngrid,d);
#{ \it xgrid } est une grille pour l'int\'egrale sur X de taille ngrid.

nombre<-nbudget-m;
for (v in 1:nombre){
  A<-nextPointMoulines(x,y);
```

```
  xn <- A$point
  yn <- f(xn);
  x <- rbind(x, xn);
  y <- c(y, yn);
  { \it xgrid }<-{ \it xgrid }[-A$indice, ];

}

print(est(y));
print(proc.time());
```

11.2 Mon programme avec formules de mise à jour

```
 library(MASS)
library(pbivnorm)
library(lhs)
library("mnormt")

#Param\‘etre de notre noyau gaussien.
l=0.15;


#On cr\’ee la norme 2.
mynorm<- function(x){
  sqrt(sum(x^2));
}

#On cr\’ee la fonction noyau (gaussienne).
kern<-function(x1, x2){
  exp(-mynorm(x1-x2)^2/(2*l^2));
}

# On cr\’ee les diff\’erentes matrices intervenant dans les formules de krigeage.
Kvec<-function(x1, x2){
  o <- dim(x1)[1];
  p <- dim(x2)[1];
  K <- matrix(0, nrow=o, ncol=p);
```

```
  for (i in 1:o){
    for (j in 1:p){
      K[i, j] <- kern(x1[i,], x2[j,]);
    }
  }
  K;
}

# On applique les formules de krigeage.

loiCond <- function(x, y, xnew){
  #Ici x sont les points d'\'evaluation, y leur \'evaluation et xnew les points o\
  res <- list();
  K <- Kvec(x,x);
  # Le m\^{e}me K que dans l'article de T\'el\'ecom.
  k <- Kvec(x, xnew);
  # Le m\^{e}me petit k que dans l'article si xnew n'a qu'un \'el\'ement.
  knew<-Kvec(xnew, xnew);

  res$esp <- t(k) %*% solve(K, y);
  #formule pour l'esp\'erance

  res$cov <- knew - t(k) %*% solve(K, k);
  res;
}


#On cr\'ee notre estimateur du quantile
est<- function (y) {
  alpha <- 0.95;
  z <- sort(y);
  z[floor(length(y) * alpha)+1];
}




#Fonction pour rechercher le point suivant, lorsque nous sommes pass\'es une fois
nextPointTatPich <- function (x,y,{ \it xsuiv },{ \it xgrid },MS,VS,MG,VG,Cov,msopt
```

```
alpha=0.95; #Param\‘etre du quantile
n<-dim(x)[1]; #Num\’ero de l’Őtape
print(n);
ngrid<-dim({ \it xgrid })[1]; #Taille de { \it xgrid }
nsuiv<-dim({ \it xsuiv })[1]; # Taille courante de { \it xsuiv }

#Pour pouvoir stocker Ĺ nouveau, on met nos stocks dans des variables "boucle" et
Covboucle<-Cov;
Coptboucle<-Copt;
vsoptboucle<-vsopt;
msboucle<-MS;
msoptboucle<-msopt;
vgboucle<-VG;
vsboucle<-VS;
mgboucle<-MG;

integrale<-array(0, nsuiv); # Vecteur dans lequel se trouverons les approximation

Cov<-matrix(0, nrow=nsuiv, ncol=ngrid);
MS<-array(0, nsuiv);
VS<-array(0, nsuiv);
MG<-matrix(0, nrow=nsuiv, ncol=ngrid);
VG<-matrix(0, nrow=nsuiv, ncol=ngrid);

for (j in 1:nsuiv){ #Boucle pour remplir integrale

  #Utilisation des formules de mises Ĺ jour pour les paramŔtres ne dŐpendant que
  moy{ \it xsuiv }<-msboucle[j]+Suivopt[j]/vsoptboucle*(yn-msoptboucle);
  var{ \it xsuiv }<-vsboucle[j]-(Suivopt[j])^2/vsoptboucle;

  contrib<-array(0, ngrid); # Vecteur dans lequel nous mettrons les contributions

  for(k in 1:ngrid){ # Boucle sur les x grid pour remplier contrib

    #Au lieu d’appeker loiCond, on utilise ici les formules de MAJ.
    cn<-Covboucle[j,k]-(Coptboucle[k]*Suivopt[j]/(vsoptboucle));
    moy{ \it xgrid }<-mgboucle[k]+Coptboucle[k]/vsoptboucle*(yn-msoptboucle);
    var{ \it xgrid }<-vgboucle[k]-(Coptboucle[k])^2/vsoptboucle;
    varfutur{ \it xgrid }<-var{ \it xgrid }-cn^2/var{ \it xsuiv };
```

```
#Et on stocke les nouvelles valeurs.
Cov[j, k]<-cn;
MG[j, k]<-moy{ \it xgrid };
VG[j, k]<-var{ \it xgrid };

#Matrices de covariance pour les fonctions bigaussiennes
beta<-cn/(sqrt(varfutur{ \it xgrid })*sqrt(var{ \it xsuiv }));
nu<-(cn-var{ \it xsuiv })/(sqrt(varfutur{ \it xgrid })*sqrt(var{ \it xsuiv }))
sigmabeta<-matrix(data=c(1, beta, beta, 1+beta^2), nrow=2);
sigmabetamoins<-matrix(data=c(1, -beta, -beta, 1+beta^2), nrow=2);
sigmanu<-matrix(data=c(1, nu, nu, 1+nu^2), nrow=2);
sigmanumoins<-matrix(data=c(1, -nu, -nu, 1+nu^2), nrow=2);

#On ordonne nos valeurs de y.
z<-sort(y);

r1<-floor(n*alpha)+1;
r2<-floor((n+1)*alpha);
if(r1==r2){ # Premi\‘ere boucle if. Premier cas : le moins courant, on ne cha

  #On calcule les fonctions de r\’epartition, attention aux cas limites.
  b1<-(z[floor(n*alpha)+1]-moy{ \it xsuiv })/sqrt(var{ \it xsuiv });
  a1<- - (z[floor(n*alpha)+1]-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid });
  phi1=pmnorm(c(b1, a1), c(0,0), sigmabetamoins);

  if(floor(n*alpha)+1==n){#Attention au cas o\‘u le quantile estim\’e est le
    b2<- - (z[floor(n*alpha)+1]-moy{ \it xsuiv })/sqrt(var{ \it xsuiv });
    znx<- - (moy{ \it xsuiv }-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid });
    phi2=pmnorm(c(b2, znx), c(0,0), sigmanu);

    contrib[k]<-abs(phi1+phi2-(1-alpha));
  }else{
    b2<-(z[floor(n*alpha)+2]-moy{ \it xsuiv })/sqrt(var{ \it xsuiv });
    znx<- -(moy{ \it xsuiv }-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid });
    phi2=pmnorm(c(b2, znx), c(0,0), sigmanumoins);

    b4<- - (z[floor(n*alpha)+2]-moy{ \it xsuiv })/(sqrt(var{ \it xsuiv }));
    a4<- - (-z[floor(n*alpha)+2]-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid ]
    phi4=pmnorm(c(b4, a4), c(0,0), sigmabeta);
```

```
      b3<-(z[floor(n*alpha)+1]-moy{ \it xsuiv })/sqrt(var{ \it xsuiv });
      znx<- - (moy{ \it xsuiv }-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid });
      phi3=pmnorm(c(b3, znx), c(0,0), sigmanumoins);

      contrib[k]<-abs(phi1+phi2-phi3+phi4-(1-alpha));
    }#Fin du deuxiŕme if.

  }#Fin du premier if
  else{#Deuxi\‘eme cas sur la partie enti\‘ere de n*alpha

    b4<- - (z[floor(n*alpha)+1]-moy{ \it xsuiv })/sqrt(var{ \it xsuiv });
    a4<- - (-z[floor(n*alpha)+1]-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid })
    phi4=pmnorm(c(b4, a4), c(0,0), sigmabeta);

    if(floor(n*alpha)+1==1){#Attention au cas oİ le quantile estim\'e est le pl

      b2<-(z[floor(n*alpha)+1]-moy{ \it xsuiv })/sqrt(var{ \it xsuiv });
      znx<- -(moy{ \it xsuiv }-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid });
      phi2=pmnorm(c(b2, znx), c(0,0), sigmanumoins);

      contrib[k]<-abs(phi2 + phi4-(1-alpha));
    }else{#Fin deuxi\‘eme if
      b1<-(z[floor(n*alpha)]-moy{ \it xsuiv })/sqrt(var{ \it xsuiv });
      a1<-(-z[floor(n*alpha)]-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid });
      phi1=pmnorm(c(b1, a1), c(0,0), sigmabetamoins);

      b2<-(z[floor(n*alpha)+1]-moy{ \it xsuiv })/sqrt(var{ \it xsuiv });
      znx<- -(moy{ \it xsuiv }-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid });
      phi2=pmnorm(c(b2, znx), c(0,0), sigmanumoins);

      b3<-(z[floor(n*alpha)]-moy{ \it xsuiv })/sqrt(var{ \it xsuiv });
      znx<- - (moy{ \it xsuiv }-moy{ \it xgrid })/sqrt(varfutur{ \it xgrid });
      phi3=pmnorm(c(b3, znx), c(0,0), sigmanumoins);

      contrib[k]<-abs(phi1+phi2-phi3+phi4-(1-alpha));
    }#Fin deuxi\‘eme if

  }#Fin du deuxi\‘eme if

}#On sort de la boucle sur les { \it xgrid } et remplie integrale et on stocke
```

```
    integrale[j]<-1/ngrid*sum(contrib);

    MS[j]<-moy{ \it xsuiv };
    VS[j]<-var{ \it xsuiv };

  }#Fin de la boucle sur les { \it xsuiv }.



  #Recherche du x_{n+1}^* et r\'ecup\'eration des valeurs stock\'ees.
  jopt<-which.min(integrale);
  lepoint<-{ \it xsuiv }[jopt,];

  list(jopt=jopt, point=lepoint, Cov=Cov, MS=MS, MG=MG[1,], VS=VS, VG=VG[1,]);
}


#Sur un exemple :
print(proc.time());
nbudget=100; #Nombre de points que l'on peut \'evaluer
mini<-20; # Nombre de points que l'on veut "sacrifier" pour initialiser

d<-2;
#d est la dimension dans laquelle on travaille.

#Fonction test
#f <- function(x) mynorm(x)^2 + sin(3*mynorm(x)*pi);
f <- function(x,y) (sin(pi*(2*x-1))+ 7*sin(pi*(2*y-1))^2 + 0.1*pi^4*sin(pi*(2*x-1)
#f <- function(v,w) ((8*v-2)*exp(-(8*v-2)^2 - (8*w-2)^2));
#f <- function(x,y) (abs(4*x-2)*abs(4*y-2));


precisionsuiv<-500; #Taille de la grille de points sur laquelle on choisira le poi
{ \it xsuiv }<-improvedLHS(precisionsuiv, d); # Cr\'eation de cette grille

#Estimation du quantile avec cette grille
ysuiv <- f({ \it xsuiv }[1,1], { \it xsuiv }[1, 2]);
for (j in 2: precisionsuiv){
  ysuiv<-c(ysuiv, f({ \it xsuiv }[j,1 ], { \it xsuiv }[j, 2]));
}
```

```
z<-sort(ysuiv);
a<-floor(precisionsuiv*0.95)+1
q1<-z[a];


nsuiv<-dim({ \it xsuiv })[1]; #Taille courante de la grille (on enleve les points d


precisiongrid<-100; # Taille de la grille { \it xgrid } servant \‘a approximer l’in
{ \it xgrid }<-improvedLHS(precisiongrid,d);
ngrid<-dim({ \it xgrid })[1];
xestim<-improvedLHS(1000, d);

#m<-mini;
m<-(mini-1); #Nombre de points que l’on donne aux d\’epart Ĺ x (un est r\’eserv\’e
x <- { \it xsuiv }[1:m,]; # Remplissage de x
{ \it xsuiv }<-{ \it xsuiv }[-(1:m), ]; # On vide { \it xsuiv } des valeurs qui ont
nsuiv<-dim({ \it xsuiv })[1]; # Mise Ĺ jour de la taille de { \it xsuiv }


#Evaluations initiales
y<-f(x[1,1], x[1, 2]);
for (i in 2:m){
  y<-c(y, f(x[i,1], x[i, 2]));
}

n<-mini; #Etape ou nombre d’\’evaluation d\’ej\‘a faites


xn<-{ \it xsuiv }[1, ]; # Faux premier point s\’electionn\’e

#Calcul des param\‘etres initiaux pour les formules de mise \‘a jour, comme si on a
xnew<-rbind({ \it xsuiv }, { \it xgrid });
loi<-loiCond(x, y, xnew);
Cov<-loi$cov[1:nsuiv, (nsuiv+1):(ngrid+nsuiv)];
MG<-loi$esp[(nsuiv+1):(ngrid+nsuiv)];
MS<-loi$esp[1 : nsuiv];
Suiv<-loi$cov[1:(nsuiv), 1:nsuiv];
VS<-diag(Suiv);
CG<-loi$cov[(nsuiv+1): (ngrid+nsuiv), (nsuiv+1): (ngrid+nsuiv)];
```

```
VG<-diag(CG);
jopt<-1;

#Evaluation
yn <- f(xn[1], xn[2]);
x <- rbind(x, xn);
y <- c(y, yn);

#Stock encore
Copt<-Cov[jopt,];
msopt<-MS[jopt];
vsopt<-VS[jopt];
Suivopt<-Suiv[jopt,];


{ \it xsuiv }<-{ \it xsuiv }[-jopt,];
Cov<-Cov[-jopt,];
MS<-MS[-jopt];
VS<-VS[-jopt];
Suivopt<-Suivopt[-jopt];
Suiv<-Suiv[-jopt,-jopt ];
nsuiv<-dim({ \it xsuiv })[1];

#Boucle sur le budget : on cherche le prochain point, on stock les valeurs utiles p
#nombre<-nbudget-m;
nombre<-nbudget-mini;
Y<-array(0, nombre); #Vecteur qui nous donne l'Ővolution de l'estimateur au cours d
X<-matrix(0, nrow=nombre, ncol=2); #Vecteurs qui nous donne les diffŐrents points j
H<-array(0, nombre)

for (v in 1:nombre){
  A<-nextPointTatPich(x,y,{ \it xsuiv },{ \it xgrid },MS,VS,MG,VG,Cov,msopt,vsopt,C
  xn<-A$point;
  yn<-f(xn[1], xn[2]);#Evaluation du point suivant
  x <- rbind(x, xn);#Mise \`a jour de x
  y <- c(y, yn);#Mise \`a jour de y
  X[v, ]<-xn;

  #Estimateur du quantile courant 1
  ba<-loiCond(x, y, xestim);
```

```
Y[v]<-est(ba$esp);

#Estimateur du Quantile courant numŐro 2
F<-mvrnorm(1000, ba$esp, ba$cov);
G<-array(0, 1000)
for (k in 1:1000){
G[k]<-est(F[k,]);}
H[v]<-mean(G);

#R\'ecup\'eration des variables de stock
Cov<-A$Cov;
MG<-A$MG;
MS<-A$MS;
VG<-A$VG;
VS<-A$VS;
jopt<-A$jopt;









#Mise \`a jour de Suiv;
nsuiv<-dim({ \it xsuiv })[1];
for(i in 1:nsuiv){
  for(j in 1:nsuiv){
    Suiv[i,j]<-Suiv[i,j]-(Suivopt[j]*Suivopt[i])/vsopt;
  }
}

#Stocks des variables utiles \`a la prochaine \'etape
Copt<-Cov[jopt,];
msopt<-MS[jopt];
vsopt<-VS[jopt];
Suivopt<-Suiv[jopt,];
Suiv<-Suiv[-jopt, -jopt];
```

```
  { \it xsuiv }<-{ \it xsuiv }[-jopt,];
  Cov<-Cov[-jopt,];
  MS<-MS[-jopt];
  VS<-VS[-jopt];
  Suivopt<-Suivopt[-jopt];

}
```

```
#Quantile empirique avec 500 observations
Z<-abs((q1-Y)/q1);
plot(Z, pch=20, type='o');
title('Erreur relative est1 (500) , l=0.15')


q2<-15.9048; #Quantile empirique avec 10^5 observations
Z<-abs((q2-Y)/q2);
plot(Z, pch=20, type='o');
title('Erreur relative est1 (10^5), l=0.15');


#Quantile empirique avec 500 observations
Z<-abs((q1-H)/q1);
plot(Z, pch=20, type='o');
title('Erreur relative est2 (500), l=0.15')


q2<-15.9048; #Quantile empirique avec 10^5 observations
Z<-abs((q2-H)/q2);
plot(Z, pch=20, type='o');
title('Erreur relative est2 (10^5), l=0.15');



u <- v <- seq(0, 1, 0.01)
w <- outer(u, v, FUN = f)
```

```
contour(u, v ,w, levels = sort(z)[floor(length(z)*0.95)])
par(new=TRUE)
plot(X, col=grey((1:nombre)/nombre), pch=20);
title('RŐpartitions des points jouŐs 1, l=0.15');


contour(u, v ,w, levels = sort(z)[floor(length(z)*0.95)])
par(new=TRUE)
plot(X, pch=20);
for(k in 1:nombre){
  text(X[k,1]+0.03, X[k, 2]+0.03, k);
}
title('RŐpartition des points jouŐs 2, l=0.15')

print(proc.time());
#On trouve enfin notre quantile !
```