MASTER 2 INGÉNIERIE STATISTIQUE

Quantification d'évènements rares à l'aide d'AutoEncodeurs Variationnels

BOUAFIA IMÈNE

15 Avril —- 13 Septembre

ENCADRANTS :
JÉRÔME MORIO (ONERA)
FRANÇOIS BACHOC (IMT)

PROFESSEURE RÉFÉRENTE :
PHILIPPE ANNE (NU)

Table des matières

Introduction

Cadre : Sécurité et fiabilité des systèmes

La sécurité des systèmes est d'une importance capitale dans le domaine des applications industrielles. Pour garantir cette sécurité, il est essentiel d'évaluer minutieusement la fiabilité des systèmes en simulation car les expérimentations physiques ne constituent pas un moyen d'évaluation suffisant en raison de diverses contraintes. Les outils de simulation numérique offrent une alternative précieuse, permettant de construire des modèles numériques afin d'examiner la réponse des systèmes à différents scénarios de défaillance.

Généralement, le comportement physique est encapsulé dans un code de calcul, noté Φ,

$$\Phi : \left| \begin{array}{ccc} \mathbb{R}^d & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \Phi(x) = y \end{array} \right.$$

qui prend en entrée un vecteur de variables X de dimension d (incluant des paramètres de conception, d'environnement, etc.) et produit une sortie Y représentant l'état du système. Le système est considéré comme défaillant si la sortie dépasse un seuil limite, c'est-à-dire si $\{Y > s\}$. On définit le domaine de défaillance D_f correspondant comme les combinaisons des variables d'entrées qui mènent à la défaillance du système :

$$D_f = \{x \in \mathbb{R}^d, y = \Phi(x) > s\}$$

Pour obtenir la sortie Y, il faut souvent résoudre des équations aux dérivées partielles via des méthodes telles que l'analyse par éléments finis. Toutefois, la complexité du code de calcul Φ rend impossible toute étude analytique. De plus, les appels au code sont généralement extrêmement coûteux et ne peuvent être effectués qu'en nombre limité. Une approche alternative pour analyser ces systèmes physiques consiste à adopter une méthode de type "boîte noire", où seules les entrées et les sorties du système sont connues, sans nécessiter de compréhension approfondie des mécanismes internes du système. L'entrée X est supposée aléatoire et il en est de même pour la sortie Y. Pour ces systèmes, une question centrale se pose en pratique :

Quelle est la **Probabilité de défaillance** $P_f = \mathbb{P}(Y > s)$ du système ?

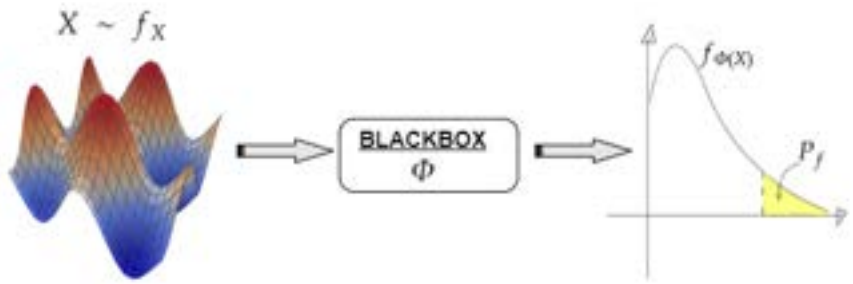


FIGURE 1 – Schématisation du contexte d'étude.

Méthodes d'estimation de la probabilité de défaillance

L'estimation de P_f n'est pas une tâche évidente, car il s'agit de l'estimation de queues de distributions. La défaillance est en effet un évènement de faible occurrence. Les méthodes classiques comme les méthodes de Monte-Carlo Naïf (MCN) sont robustes et sans biais, mais nécessitent une quantité importante de données pour obtenir une estimation précise de P_f. Cette précision est souvent indispensable dans des applications de fiabilité ou de certification.

Il existe d'autres approches probabilistes dans la littérature, telles que les Subset Simulation (SS) [1] ou l'Échantillonnage préférentiel (IS) [6]. Ces méthodes améliorent la précision de l'estimation de P_f en générant des échantillons d'entrée dans la région de défaillance D_f, tout en maintenant un budget de simulation raisonnable par rapport aux MCN.

Dans le cadre de ce stage, nous nous sommes concentrés sur la méthode des SS. Cet algorithme consiste à décomposer un événement rare (la défaillance) en une série d'événements moins rares. Plutôt que de simuler directement un événement de faible occurrence, on simule une séquence d'événements ayant des probabilités plus élevées. L'estimation de ces probabilités repose sur des méthodes d'échantillonnage telles que les méthodes de Monte Carlo Markov Chain (MCMC). Cependant, ces méthodes sont confrontées à la malédiction de la dimension, ce qui pose des problèmes de convergence, affecte la qualité des estimations, et entraîne une augmentation des coûts de simulation (c'est-à-dire des appels à Φ). Des variantes spécifiques de MCMC pour le SS en grande dimension (d grand) ont été développées, mais elles sont souvent appliquées dans des contextes restrictifs. Par exemple, l'algorithme Modified Metropolis Algorithm (MMA), proposé dans [17], nécessite des composantes d'entrée X indépendantes pour être applicable en grande dimension. Des approches où des transformations isoprobabilistes des entrées vers l'espace gaussien standard sont aussi proposées dans [11].

Objectif du stage

Dans le contexte de ce stage, nous nous concentrons sur le développement d'échantillonneurs non paramétriques efficaces pour des problèmes d'estimation d'évènement rares en grande dimension, où l' entrée X suit une distribution complexe et où l'indépendance des composantes de X n'est pas toujours garantie. Il s'agit de fiabiliser les estimations des probabilités de défaillance par SS en réduisant les corrélations au sein des échantillons générés par les algorithmes MCMC et en augmentant le taux d'acceptation. La résolution de cette tâche est particulièrement complexe en raison de la difficulté à surmonter le fléau de la grande dimension, d'autant plus que les distributions concernées sont potentiellement multimodales.

Ce rapport est structuré en trois grande parties. Dans un premier temps, nous présenterons les méthodes préexistantes dans la littérature pour l'estimation des évènements rares et plus précisément ce qui existe autour des méthodes SS. Dans un second temps, on s'intéressera à un modèle génératif, les Auto-Encodeur Variationnel (VAE) et plus particulièrement à la capacité de ces objets à échantillonner selon les lois apprises. Enfin dans une troisième et dernière partie, nous proposerons un algorithme imbriquant SS et VAE.

Dans la suite, on considère le vecteur aléatoire X de dimension d et de densité f_X.
La quantité d'intérêt à estimer est :

$$P_f = \mathbb{E}[\mathbf{1}_{\{\Phi(\mathbf{X})>s\}}] = \int_{D_f} f_X(x)\,dx = \int \mathbf{1}_{\{\Phi(x)>s\}} f_X(x)\,dx \tag{1}$$

On rappelle la définition de l'événement de défaillance :

$$D_f = \{x \in \mathbb{R}^d, y = \Phi(x) > s\} \tag{2}$$

Acronymes

ACP Analyse en Composantes Principales. 22
AE Auto-Encodeur. 22, 23, 29

CV Coefficient de Variation. 7, 19, 36

ELBO Evidence Lower BOund. 25–27, 29, 30, 33

IS Échantillonnage préférentiel. 4

M-H Metropolis-Hastings. 8, 10, 33–35, 38, 39
MCMC Monte Carlo Markov Chain. 4, 10, 19, 41
MCN Monte-Carlo Naïf. 4, 6, 10, 15, 33, 37, 40, 46, 47
MMA Modified Metropolis Algorithm. 4, 15, 17–19
MoG Mélange de gaussiennes. 26, 27, 29, 32

SIS Sequential Importance Sampling. 41
SS Subset Simulation. 1, 4, 10–16, 19, 20, 32, 33, 36, 37, 39–41, 43–47
SVD Singular Value Decomposition. 22

VAE Auto-Encodeur Variationnel. 4, 24–37, 39–41, 46, 47
VP VampPrior. 27–32, 35, 39, 40, 43–45

Chapitre 1

Revue autour des Subset Simulations pour l'estimation d'évènement rare

I Méthode de Monte-Carlo Naïve (MCN)

La méthode d'estimation de MCN [10] a été l'une des premières méthodes de simulation afin d'estimer des espérances.
Soit $X_1, \ldots, X_n$ une suite de variables aléatoires de loi f_X. On définit l'estimateur MCN de P_f comme suit :

$$\hat{P}_f^{MCN} = \frac{1}{N} \sum_{i=0}^{N} 1_{\Phi(X^{(i)})>s} \tag{1.1}$$

Par Loi Forte des Grands Nombres (LFGN), $\hat{P}_f^{MCN}$ est un estimateur sans biais de P_f et fortement consistant

$$\hat{P}_f^{MCN} \xrightarrow[N\to\infty]{ps} P_f \tag{1.2}$$

On suppose $\phi(X)$ de carré intégrable. Le théorème centrale limite sur $\hat{P}_f^{MCN}$ permet d'obtenir l'intervalle de confiance de niveau $(1-\alpha)$ suivant :

$$\begin{aligned} IC(1-\alpha) &= \left[\hat{P}_f^{MCN} \pm q_{1-\alpha/2} \frac{\sqrt{\hat{P}_f^{MCN}\left(1-\hat{P}_f^{MCN}\right)}}{\sqrt{N}} \right] \\ &= \hat{P}_f^{MCN} \left[1 \pm q_{1-\alpha/2}\, \epsilon_{rel} \right] \end{aligned}$$

Où

$$\epsilon_{rel} = \frac{\sqrt{(1-\hat{P}_f^{MCN})}}{\sqrt{N}\sqrt{\hat{P}_f^{MCN}}} \quad \Leftrightarrow \quad \epsilon_{rel}^2 = \frac{\left(1-\hat{P}_f^{MCN}\right)}{N\hat{P}_f} \tag{1.3}$$

ϵ_{rel} est l'erreur relative tolérée au sein de l'intervalle de confiance. Cette quantité dépend du nombre d'échantillons N. Sachant que P_f est une probabilité d'évènement rare et par conséquent $(1-P_f) \approx 1$, on peut estimer l'ordre de grandeur de N pour une erreur relative donnée :

$$N \propto \frac{1}{\epsilon_{rel}^2 \hat{P}_f^{MCN}} \tag{1.4}$$

Pour estimer une probabilité P_f de l'ordre 10^{-r} avec une précision de $\epsilon_{rel} = 10^{-1}$, il faut donc $N = 10^{r+2}$ réalisations de $\phi(X)$. Ceci rend cette méthode difficilement applicable sur une fonction ϕ complexe et coûteuse en temps de calcul pour l'estimation d'évènement rare.

Definition 1 On définit le Coefficient de Variation (CV) pour un estimateur quelconque $\hat{P}_f$ comme suit :

$$\text{CV} = \frac{\sqrt{\mathbb{V}(\hat{P}_f^{MCN})}}{\mathbb{E}(\hat{P}_f^{MCN})} \tag{1.5}$$

Souvent exprimé en pourcentage, le CV permet de rendre compte de la précision de l'estimateur de $\hat{P}_f$.

Dans le cas de la MCN, le CV est égal à $CV(\hat{P}_f^{MCN}) = \frac{\sqrt{(1-P_f)}}{\sqrt{P_f}\sqrt{N}}$

L'estimateur de Monte-Carlo Naïf est applicable à condition que l'on soit capable de simuler selon la loi f_X. Cependant, cela n'est pas toujours possible. Il existe diverses techniques pour obtenir des échantillons distribués conformément à la loi souhaitée. Lorsque la densité f_X est connue à une constante multiplicative près, les algorithmes MCMC définis à la section suivante sont particulièrement bien adaptés à ce contexte.

II Monte Carlo Markov Chain (MCMC)

Dans les algorithmes MCMC, l'objectif est de construire une chaîne de Markov stationnaire $\{X_t, t\}$, de mesure invariante égale à la loi cible π connue à une constante près.

II.1 Propriétés sur les chaînes de Markov

Les chaînes de Markov constituent un exemple de suite de variables aléatoires $(\mathbf{X}_t)_t$ (ou processus aléatoire pour le cas continu) à valeurs dans E, appelé *espace d'état*.

Definition 2 Une chaîne de Markov est une suite de variables $(X_n)_n$ dans un ensemble E continu de $\mathbb{R}^d$, si et seulement si pour tout k et pour tout $(x_0, \ldots, x_k)$ dans E tel que $g(x_0, \ldots, x_k) > 0$ avec g densité de la loi conjointe des k variables, la propriété suivante est vérifiée

$$f_{X_{k+1}|X_0=x_0,\ldots,X_k=x_k}(x) = f_{X_{k+1}|X_k=x_k}(x) \quad \forall x \in E.$$

On définit la probabilité de transition comme suit $q(x \mid y) = f_{X_{k+1}|\{X_k=y\}}(x)$. Cette densité est ce qu'on appelle le noyau de transition.

Definition 3 On dit qu'une chaîne de Markov $\{X_t, t\}$ avec un noyau de transition T satisfait la condition d'irréversibilité s'il existe une fonction π satisfaisant l'égalité qui suit :

$$\pi(x)T(y \mid x) = \pi(y)T(x \mid y) \quad x, y \in \mathbb{R}^p.$$

Theorem 1 *On suppose que la chaîne de Markov a un noyau de transition T qui vérifie la condition d'irréversibilité pour une certaine densité π. Dans ce cas, π est la loi invariante de la chaîne.*

La condition d'irréversibilité est donc une condition suffisante pour converger vers la loi cible π.

II.2 Algorithme Metropolis-Hastings (M-H)

On se donne un noyau de Markov Q, appelé *noyau de proposition* et on construit une chaîne de Markov $(X_t)_t$ par l'algorithme 1 [13].

Algorithm 1 Algorithme Metropolis-Hastings

Require: — Noyau de proposition Q
— M longueur de la chaîne
— initialisation $X_0 = x_0$ telle que $\pi(x_0) > 0$
while $i < M$ **do**
À partir de l'état $X_i = x_i$, on propose $\tilde{x} \sim Q(. \mid x_i)$
On accepte $\tilde{x}$ avec une probabilité

$$\alpha(\tilde{x}, x_i) = min\left\{1, \frac{\pi(\tilde{x})Q(x_i \mid \tilde{x})}{\pi(x_i)Q(\tilde{x} \mid x_i)}\right\}$$

Par conséquent,

$$x_{i+1} = \begin{cases} \tilde{x} & \text{if } u < \alpha(x_i, \tilde{x}) \\ x_i & \text{if } u \geq \alpha(x_i, \tilde{x}) \end{cases} \qquad u \sim \mathcal{U}[0,1]$$

$i = i + 1$
end while
return $\{X_i, i = 0, \dots M\}$

On définit le support d'un noyau de proposition comme suit

$$supp(Q) = \bigcup_{x \in E} supp(Q(. \mid x).$$

Theorem 2 *Soit $\{X_t, t\}$ la chaîne de Markov renvoyée par l'algorithme Metropolis-Hastings (M-H). Quel que soit le noyau de proposition Q dont le support inclut celui de la densité π, $supp(Q) \supset supp(\pi)$ on a*

1. *Le noyau de transition de la chaîne vérifie la condition d'irréversibilité pour π.*
2. *π est la loi stationnaire de la chaîne.*

Il existe différentes manières de sélectionner le noyau de proposition pour l'algorithme M-H. Cependant, dans notre étude, nous nous limitons à deux types d'algorithmes : le Metropolis-Hastings indépendant et le Metropolis-Hastings avec marche aléatoire, décrits comme suit :

$$\mathbf{X}^* \sim Q \qquad \mathbf{X}^* \text{ est indépendant de } \mathbf{X}_t$$
$$\mathbf{X}^* = \mathbf{X}_t + \epsilon_t \qquad \epsilon_t \overset{iid}{\sim} q.$$

Remark 1 Un bon noyau de proposition pour l'algorithme M-H est la densité π.

$$Q(. \mid x) = \pi(.) \quad \forall x \in E. \tag{1.6}$$

Avec Q, le noyau de proposition. Cependant, cette alternative est difficile à satisfaire puisqu'on ne connaît pas l'expression explicite de π et que l'on ne sait pas échantillonner selon π. Par contre, on peut espérer qu'un noyau Q proche de π en un certain sens soit également un bon noyau de proposition.

II.3 Qualité de simulation

Pour attester de la qualité des chaînes renvoyées par l'algorithme M-H, on s'intéresse à plusieurs propriétés. On peut se demander si on explore tout l'espace état E mais surtout comment on l'explore.

Definition 4 (Taux d'acceptation) Soit $\{X_t, t = 0,\dots,N\}$, la chaîne de Markov obtenue par algorithme M-H. Le taux d'acceptation est donné par

$$\rho = \frac{1}{N}\sum_{t=1}^{N} \mathbf{1}_{\{X_{t-1}\neq X_t\}}.$$

Cet indicateur permet d'évaluer la capacité du noyau de proposition à générer des valeurs dans les régions pertinentes, c'est-à-dire les zones où la densité de la distribution cible (selon laquelle on aimerait échantillonner) présente des masses de probabilité élevées. Toutefois, en cherchant un fort taux d'acceptation, nous avons

$$\mathbb{P}(||\,\mathbf{X}^* - \mathbf{X}_t\,||> \epsilon \mid \mathbf{X}_t = \mathbf{x}_t) \ll 1.$$

Ceci induira une lente exploration de l'espace d'état.

On peut estimer également l'autocorrélation, Celle-ci mesure la dépendance (linéaire) entre les valeurs de la chaîne.

Definition 5 (Autocorrélation) On suppose que la chaîne $\{X_t, t = 0,\dots,N\}$ appartient à L^2 alors la fonction d'autocovariance de cette chaîne est définie comme suit :

$$\gamma(t,s) = Cov(X_t, X_s) = \mathbb{E}[(X_t - \mu_t)(X_s - \mu_s)]$$

avec $\mu_t = \mathbb{E}(X_t)$.
Ceci induit la définition de la fonction d'autocorrélation :

$$\rho(t,s) = \frac{\gamma(t,s)}{\gamma(t,t)\gamma(s,s)}$$

Une faible autocorrélation indique que les échantillons successifs sont presque indépendants, ce qui peut suggérer que la chaîne explore efficacement l'espace E. Cependant, cela peut également entraîner un taux d'acceptation très faible, ce qui signifie que les transitions proposées sont peut-être trop ambitieuses. Inversement, un taux d'acceptation élevé peut induire une forte corrélation entre les échantillons, ralentissant ainsi l'exploration de l'espace, comme mentionné précédemment.

Definition 6 (Loi stationnaire) La loi stationnaire (ou distribution stationnaire) d'une chaîne de Markov est une distribution de probabilité π sur l'espace d'états E telle que, si la chaîne commence dans cette distribution, elle reste dans cette distribution à chaque étape de son évolution.
Formellement, cette loi stationnaire satisfait aux equations d'équilibre

$$\pi(y) = \int_E \pi(x)T(y \mid x)dx \qquad \forall\, y \in E$$

Definition 7 (chaîne de Markov ergodique) Une chaîne de Markov $\{X_t, t\}$ est dite ergodique si elle est irréductible et positive récurrente. Soit T, le noyau de transition, les conditions d'ergodicité peuvent être formulées comme suit :

1. **Irréductibilité** : Pour tout $x, y \in E$, il existe une probabilité non nulle de passer de x à y, i.e qu'il existe un chemin de transition $x = x_0, x_1, \dots, x_k = y$ tel que

$$T(x_1 \mid x_0) > 0, \dots, T(y \mid x_{k-1}) > 0.$$

2. **Récurrence positive** Soit $\mathrm{A} \subset E$.
On pose $T_\mathrm{A} := inf\{n \leq 0,\ X_n \in \mathrm{A}\}$, alors

$$\mathbb{P}\Big(T_\mathrm{A} < \infty \mid X_0 = x\Big) = 1 \quad \forall\, x \in E$$

Une chaîne ergodique garantit que, indépendamment de l'état de départ, la chaîne va explorer tout l'espace d'état. En d'autres termes, si une chaîne de Markov est ergodique, on peut être certain que la distribution de la chaîne finira par se stabiliser vers la distribution stationnaire.

Après avoir introduit les deux outils nécessaires à la définition de la méthode SS, nous allons maintenant décrire en détail son processus de fonctionnement et en discuter les implications.

III Subset Simulation

III.1 Principe Général

La méthode des Subset Simulations est une méthode adaptative qui permet d'estimer la probabilité de défaillance P_f. Dans l'algorithme SS, on considère une suite d'évènements décroissants au sens de l'inclusion de telle sorte qu'on puisse réécrire l'évènement de défaillance comme leur intersection :

$$F_1 \supset F_2 \supset \cdots \supset F_m = D_f \text{ et } D_f = \bigcap_{i=1}^{m} F_i$$

Par un conditionnement successif, on obtient une nouvelle expression de la probabilité de défaillance :

$$P_f = \mathbb{P}(F_0) \prod_{i=1}^{m-1} \mathbb{P}(F_{i+1} \mid F_i) \tag{1.7}$$

La motivation est, qu'en choisissant les évènements $(F_i)_i$ de façon appropriée, les probabilités conditionnelles dans (1.7) seront assez larges pour être estimé facilement par MCN.

Le découpage en m problèmes de défaillance intermédiaires induit l'apparition de lois conditionnelles.

$$f_{X|F_j}(x) = \frac{f_X(x)\mathbf{1}_{F_j}(x)}{P(F_j)}, \quad j \in \{0, \ldots, m-1\}. \tag{1.8}$$

Ces distributions sont connues à une constante près.

La quantité $\mathbb{P}(F_0)$ est estimée par MCN

$$\hat{P}_0 = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{F_0}(X^{(i)})$$

Cependant, les $\hat{P}_j = \mathbb{P}(F_j \mid F_{j-1})$ ne peuvent être estimés par une méthode MCN à l'aide des échantillons de départ $(X^{(i)})_{i=1}^{N} \overset{i.i.d}{\sim} f_X$ au risque d'être moins précis que pour P_0. Possédant les expressions des densités dans l'équation (1.8) à une constante près, on peut échantillonner par un algorithme MCMC, tel que décrit par l'algorithme 1 (M-H) puis procéder à l'estimation de P_j par MCN à l'aide de ces nouveaux échantillons.
Formellement, soit $\{X^{(i),(j)}, i \in [\![1, N]\!]\}$, chaîne renvoyée par l'algorithme M-H pour la loi $f_{X|F_j}(x)$, alors on a :

$$\hat{P}_j = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{F_0}(X^{(i),(j)})$$

Remark 2 L'algorithme est connu sous le nom de *Subset Simulation*, mais a également été étudié dans la littérature mathématique sous le nom de *Sequential Monte Carlo* [3].

III.2 Limites d'implémentation de la méthode

Lors de la mise en œuvre de l'algorithme SS, on se heurte à plusieurs difficultés.

Choix des seuils intermédiaires

Le choix des événements intermédiaires $(F_i)_i$ joue un rôle crucial dans l'obtention d'une bonne estimation de P_f par une méthode SS. En effet, on pourrait considérer une suite de seuils croissants $s_0 < \cdots < s_m = s$ et ceci conduit à une représentation des évènements comme suit :

$$F_i = \{x \in \mathbb{R}^d,\ \Phi(x) > s_i\} \quad \forall i \in \{1, \ldots, m\}.$$

La limite d'un telle procédure est qu'on ignore quelles valeurs de seuils prendre a priori. Prendre une suite de seuils qui croît lentement entraînerait l'apparition de probabilités conditionnelles $(P_j)_j$ grandes, conduisant à moins d'échantillons nécessaires pour leur estimation. Cependant, cette approche requiert un nombre d'événements m plus importants. À l'inverse, une séquence qui croît rapidement entraînera l'apparition de phénomènes plus rares et donc des probabilités $(P_j)_j$ plus faibles. Ce scénario exige plus d'échantillons à chaque itération pour une estimation plus précise.

Une alternative est de fixer une valeur de probabilité conditionnelle, p_0 et de construire les seuils de manière adaptative, comme expliquée dans l'algorithme 2. De cette façon, on se dirige vers les zones de défaillances de manière adaptative tout en gardant un ordre de grandeur raisonnable pour les $(P_j)_j$.

L'exemple bidimensionnel en Figure 1.1 suivant permet d'illustrer le comportement du SS et de mieux comprendre son fonctionnement.

$$\begin{cases} \Phi(x) = x_1^2 + x_2^2 & x \in \mathbb{R}^2 \\ \Phi(\mathbf{X}) \sim \chi^2 & \mathbf{X} \sim \mathcal{N}(0, I_2). \end{cases}$$

On considère l'évènement de défaillance suivant $D_f = \{\Phi(\mathbf{X}) > 10\}$ et sa probabilité de défaillance est $P_f = 0.006738$

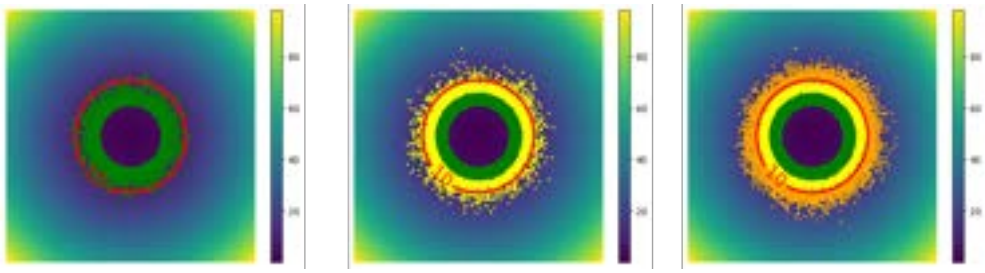
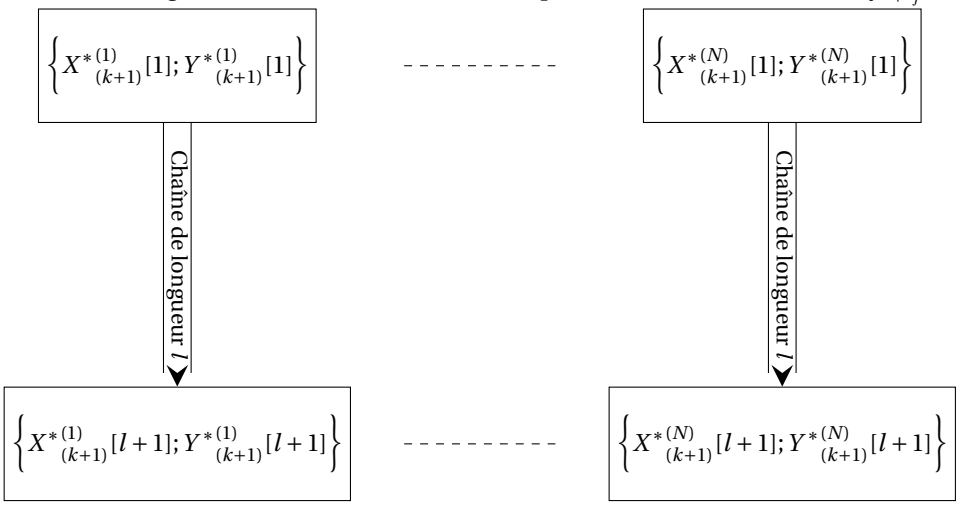


FIGURE 1.1 – Illustration du comportement de l'algorithme 2. Les échantillons renvoyés, appartenant aux évènements F_1, F_2 et F_3 sont pour un seuil de probabilité fixe $p_0 = 0.1$. Les échantillons de l'évènement F_3 représentent les échantillons de l'évènement de défaillance car ils dépassent le seuil fixe 10.

Algorithm 2 Vanilla Subset Simulation

Require: — Seuil probabilité fixe p_0
— N-échantillon de départ $\{X^{(1)},\dots,X^{(N)}\}$
— $\{Y^{(1)},\dots,Y^{(N)}\} = \Phi(X^{(1)}),\dots,\Phi(X^{(N)})$
— Seuil limite s
— Noyau de proposition Q

1: On pose $\overleftarrow{F}_N^{(k)}$ processus quantile de l'échantillon $E_{(k)} = \{Y_{(k)}^{(1)},\dots,Y_{(k)}^{(N)}\}$, et $\hat{\gamma}_N^{(k)} = \overleftarrow{F}_N^{(k)}(1-p_0)$ le seuil considéré au k^e événement et $A_{(k)} = \{X_{(k)}^{(1)},\dots,X_{(k)}^{(N)}\}$, l'échantillon associé.
2: $k = 1$
3: **while** $\hat{\gamma}_N^{(k)} < s$ **do**
4: On considère $\tilde{A}_{(k)} = \{X_{(k)}^{(i)}, \quad Y_{(k)}^{(i)} > \hat{\gamma}_N^{(k)} \ \forall i\}$, par définition de $\hat{\gamma}_N^{(k)}$, $card\left(\tilde{A}_{(k)}\right) = \lfloor Np_0 \rfloor$
5: On tire uniformément avec remise dans $\tilde{A}_{(k)}$ N échantillons et on obtient $A^*_{(k)} = \{X^{*(1)}_{(k+1)},\dots,X^{*(N)}_{(k+1)}\}$
6: Application de N algorithmes **M-H** dont chaque initialisation est un élément de $A^*_{(k)}$. On effectue des chaînes de longueur l et la loi stationnaire créée par les chaînes de Markov est $f_{X|F_j}$.



7: $k = k+1$
8: On pose $A_{(k)} = \{X_{(k)}^{(1)}[l+1],\dots,X_{(k)}^{(N)}[l+1]\}$, $E_{(k)} = \{Y_{(k)}^{(1)}[l+1],\dots,Y_{(k)}^{(N)}[l+1]\}$ et $\hat{\gamma}_{(N)}(k)$ associé
9: **end while**
10: **return** $\hat{P}_f^{SS} = p_0^{k-1}\dfrac{card\left\{Y_{(k)}^{(i)} > s,\ Y_{(k)}^{(i)} \in E_{(k)}\right\}}{N}$

Ergodicité dans le cadre des Subset Simulation

La deuxième problématique à laquelle on se heurte est l'ergodicité des chaînes de Markov $\{X_t, t\}$ renvoyées par la méthode SS pour l'estimation des probabilités $(P_j)_{j=1}^m$.

En pratique, la méthode SS rencontre des défis puisqu'il s'agit d'une taille de chaîne finie. Dans ce cas de figure, il est plus difficile d'atteindre l'ergodicité, encore plus si les régions de défaillances sont non connexes dans l'espace d'état. Ces limitations peuvent renvoyer de moins bons estimateurs $\hat{P}_f^{SS}$.

1. Estimation biaisée : Lorsque certaines régions contribuant à la défaillance sont rarement ou jamais visitées pendant la simulation, l'estimateur $\hat{P}_f^{SS}$ devient biaisé. Ce biais provient de l'exploration limitée de l'espace d'état, conduisant à une sous-représentation des régions critiques de défaillance.
2. Difficultés d'exploration : Les régions de défaillance non continues peuvent poser des défis importants pour l'exploration de l'espace d'état. Si le noyau de proposition, Q, manque de capacité d'étalement suffisante par rapport à la taille des zones non défaillantes, il peut avoir

du mal à passer d'une zone de défaillance à une autre. Cela peut conduire à un regroupement des échantillons dans une seule zone défaillante. La figure 1.2 illustre ce phénomène.

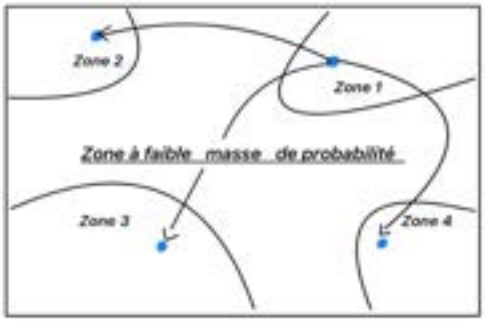


FIGURE 1.2 – Illustration d'un exemple en 2D où la zone de défaillance est une union de plusieurs zones (ici 4) $D_f = \cup_{i=1}^{4}$ Zone i. La difficulté ici est le passage d'une zone à une autre zone afin d'explorer l'ensemble des zones de défaillances.

Pour remédier à ces limitations et améliorer les performances en pratique, la stratégie suivante peut être envisagée ; considérer plusieurs chaînes avec différents états initiaux obtenus à partir de l'évènement précédent F_j (voir le schéma contenu dans l'algorithme 2 à la ligne 6). Ces initialisations, tirées de différentes régions de défaillance, peuvent guider les chaînes vers divers modes de défaillance, améliorant ainsi la couverture globale de l'espace d'état. L'efficacité de cette méthode reposerait tout d'abord sur l'estimation de P_0. Avec une bonne estimation $\hat{P}_0$ i.e un coefficient de variation faible, on peut s'attendre à une meilleure répartition des échantillons dans les régions de défaillance associées à différents événements.

Il est important de noter que les stratégies d'atténuation proposées sont valides sous l'hypothèse que les régions ne contribuant pas à la défaillance à de bas seuils conditionnels, s_j, restent inactives à des seuils conditionnels plus élevés.

III.3 Algorithmes MCMC (M-H) dans le cas des Subset Simulation

Dans le cadre des SS, l'expression de la probabilité d'acceptation pour $x_i \in F_j$ est la suivante :

$$\alpha(\tilde{x}, x_i) = min\left\{1, \frac{f_X(\tilde{x})Q(x_i \mid \tilde{x})}{f_X(x_i)Q(\tilde{x} \mid x_i)} \mathbf{1}_{F_j}(\tilde{x})\right\} \tag{1.9}$$

$$= a(\tilde{x}, x_i)\mathbf{1}_{F_j}(\tilde{x}) \tag{1.10}$$

où $a(\tilde{x}, x_i) = \min\left\{1, \frac{f_X(\tilde{x})Q(x_i|\tilde{x})}{f_X(x_i)Q(\tilde{x}|x_i)}\right\}$.

L'algorithme peut être alors décomposé en 2 parties :

1. Accepter ou rejeter la proposition $\tilde{x}$: $v = \begin{cases} \tilde{x} & \text{avec probabilité } a(\tilde{x}, x_i) \\ x_i & \text{avec probabilité } 1 - a(\tilde{x}, x_i). \end{cases}$
2. Accepter ou rejeter v : $x_{i+1} = \begin{cases} v, & v \in F_j \\ x_i, & v \notin F_j. \end{cases}$

Cette approche est intéressante, car l'évaluation de la fonction Φ en $\tilde{\mathbf{x}}$ peut-être faite seulement après acceptation avec probabilité $a(\tilde{x}, x_i)$. En effet, il se peut que les propositions $\tilde{x}$ ne soient pas satisfaisantes pour la distribution de départ f_X et donc il serait inutile de faire appel au code Φ pour vérifier l'appartenance à F_j. Ceci permettra une économie du budget.

> **Remark 3 (Limites en grande dimension)** Il est plus difficile d'explorer l'espace état, dans un domaine de plus grande dimension, car l'indicatrice $\mathbf{1}_{F_j}$ engendre très souvent du rejet. Une illustration du fort rejet des MCMC en grande dimension dans l'algorithme SS est faite dans [11] avec l'exemple du noyau gaussien.

Maintenant que nous avons exploré en détail la méthode SS, nous allons mettre en place un test pour évaluer son comportement dans différentes conditions, telles qu'un domaine de défaillance non connexe ou une dimension d du vecteur élevée.

IV Exemples Tests

La fonction *4-branches* est un problème de référence largement utilisée dans les analyses de fiabilité.

$$\Phi(x) = -\min \begin{cases} \beta + \frac{1}{\sqrt{d}} \sum_{i=1}^{d} x_i = \beta + t_1 \\ \beta - \frac{1}{\sqrt{d}} \sum_{i=1}^{d} x_i = \beta - t_1 \\ \beta + \frac{1}{\sqrt{d}} \left(\sum_{i=1}^{\lfloor d/2 \rfloor} x_i - \sum_{i=1+\lfloor d/2 \rfloor}^{d} x_i \right) = \beta + t_2 \\ \beta + \frac{1}{\sqrt{d}} \left(-\sum_{i=1}^{\lfloor d/2 \rfloor} x_i + \sum_{i=1+\lfloor d/2 \rfloor}^{d} x_i \right) = \beta - t_2. \end{cases}$$

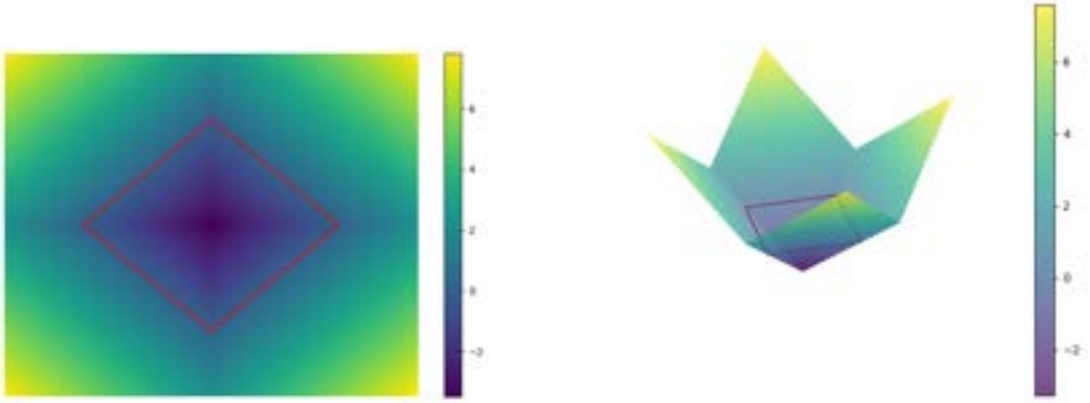
Cette exemple-test définit une série de systèmes linéaires possédant 4 zones de défaillances. Il prend en entrée un vecteur gaussien $\mathbf{X} \sim \mathcal{N}(0, I_d)$ vivant dans $\mathbb{R}^d$ et a pour particularité d'avoir une probabilité de défaillance indépendante de la dimension d.
En effet, comme $\beta \pm t_i \sim \mathcal{N}(\beta, 1), \quad i = 1,2$ et $\text{cov}(t_1, t_2) = 0$ (démontrée en annexe), la distribution de $\Phi(\mathbf{X}) = -\min\{\beta \pm t_i, \quad i = 1, \ldots, 4\}$ est indépendante de d.

On peut alors s'intéresser à l'événement de défaillance suivant :

$$P_f = \mathbb{P}(\Phi(X) \geq 0) \approx 4\phi_{\mathcal{N}(0,1)}(-\beta) \tag{1.11}$$

Les paramètres choisis pour cet exemple sont les suivants :

$$\begin{array}{llll} \beta = & 3.5 & P_f = & 9.3 \times 10^{-4} \\ \beta = & 5 & P_f = & 1.15 \times 10^{-6}. \end{array}$$



(a) Représentation dans le plan. **(b)** Représentation dans l'espace.

FIGURE 1.3 – Graphique du comportement de la fonction 4-branches Φ pour le cas de la dimension $d = 2$

IV.1 Application de l'algorithme SS

On utilise la méthode de SS pour estimer les probabilités de défaillance de l'exemple précédent. Etant donné que les entrées sont gaussiennes, nous choisissons une marche aléatoire dans l'algorithme M-H où la perturbation est

$$\epsilon_t \sim \mathcal{N}(o, I_d \times \sigma).$$

Afin de pouvoir comparer les performances entre estimateurs de P_f, on introduit la mesure

suivante :

$$\nu = \frac{N_{req}}{N_{tot}} \quad \text{avec} \quad N_{req} = \frac{1-P_f}{P_f cov(\hat{P_f}^{SS})^2} \quad \text{et} \quad N_{tot}, \text{ nombre total d'appel à la fonction } \Phi.$$

Un coefficient $\nu < 1$ signifiera qu'une simple méthode MCN suffira pour obtenir une précision égale et à moindre coût. Tandis qu'un coefficient $\nu > 1$ démontrera qu'il est plus intéressant de procéder par une méthode SS.

Les propriètés (moyennes, variances ...) des estimateurs issus des algorithmes 2 et MMA sont obtenues à partir de 100 itérations.

Application de l'algorithme SS pour le cas : $\beta = 3.5 \quad P_f = 9.3 \times 10^{-4}$

Dans ce cas, la probabilité P_f est approximativement de l'ordre de 10^{-3}. L'algorithme SS est plus adapté pour des probabilités inférieures à 10^{-4} comme cela est couramment souligné dans la littérature ; par conséquent, les performances évaluées à travers le paramètre ν ne montrent pas de différences significatives par rapport à une méthode MCN. Les résultats sont donnés en Table 1.1.

Dimension	$\hat{P_f}^{SS}$	CV	ν	σ
$d = 5$	9.33×10^{-4}	0.056	1.35	0.4
$d = 10$	9.31×10^{-4}	0.066	1.13	0.4
$d = 50$	9.29×10^{-4}	0.13	0.27	0.3
$d = 100$	9.39×10^{-4}	0.19	0.13	0.2

(a) Résumé de la qualité de l'estimateur P_f^{SS}

Évènement	F_0	F_1	F_2	F_3	F_4
$d = 5$	0.40	0.35	0.33	0.31	0.28
$d = 10$	0.33	0.29	0.27	0.25	0.24
$d = 50$	0.20	0.19	0.17	0.16	0.15
$d = 100$	0.23	0.22	0.21	0.20	0.19

(b) Taux d'acceptation ρ (définit dans (4)) pour chaque événement

TABLE 1.1 – Estimation par l'algorithme 2 pour l'événement où $\beta = 3.5$ et $P_f = 9.3 \times 10^{-4}$. La probabilité fixe $p_0 = 0.25$. Le noyau de proposition est une marche aléatoire gaussienne. Les longueurs de chaînes sont de taille 6. Pour les taux d'acceptation ρ, ce sont les taux moyennés sur l'ensemble des chaînes pour chaque événement F_i

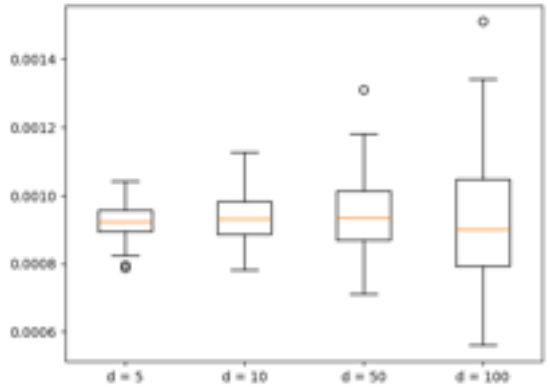


FIGURE 1.4 – Boite à moustache pour les différents estimateurs P_f^{SS} correspondant à chaque dimension d pour le cas $\beta = 3.5$ et $P_f = 9.3 \times 10^{-4}$.

On remarque que lorsqu'on augmente la dimension, il devient plus difficile d'explorer l'espace d'état et le coefficient de variation $cov(\hat{P_f}^{SS})$ croît, rendant la méthode moins précise. On peut alors choisir d'utiliser des agitations σ moins grandes, mais cela a un coût : il faut des chaînes plus longues pour atteindre la stationnarité.

La problématique des chaînes corrélées n'est pas abordée ici, car, comme expliqué dans l'algorithme 2, de faibles valeurs d'auto-corrélations ont été renvoyées par les chaînes.

Application de l'algorithme SS pour le cas : $\beta = 5 \quad P_f = 1.15 \times 10^{-6}$

Dimension	$\hat{P_f}^{SS}$	CV	ν	σ
$d = 5$	1.14×10^{-6}	0.11	127.44	0.4
$d = 10$	1.16×10^{-6}	0.11	111.4	0.4
$d = 50$	1.16×10^{-6}	0.54	4.66	0.3
$d = 100$	1.39×10^{-6}	0.19	0.13	0.2

(a) Résumé de la qualité de l'estimateur P_f^{SS}

Évènement	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
$d = 5$	0.40	0.35	0.33	0.31	0.29	0.28	0.26	0.24	0.24
$d = 10$	0.33	0.29	0.27	0.25	0.24	0.23	0.22	0.21	0.20
$d = 50$	0.20	0.19	0.17	0.16	0.16	0.15	0.15	0.14	0.14
$d = 100$	0.23	0.22	0.21	0.20	0.19	0.19	0.19	0.18	0.17

(b) Taux d'acceptation ρ pour chaque événement

TABLE 1.2 – Estimation par l'algorithme 2 pour l'événement où $\beta = 5$ et $P_f = 1.5 \times 10^{-6}$. La probabilité fixe $p_0 = 0.25$. Le noyau de proposition est une marche aléatoire gaussienne. Les longueurs de chaînes sont de taille 6. Pour les taux d'acceptation ρ, ce sont les taux moyennés sur l'ensemble des chaînes pour chaque événement F_i

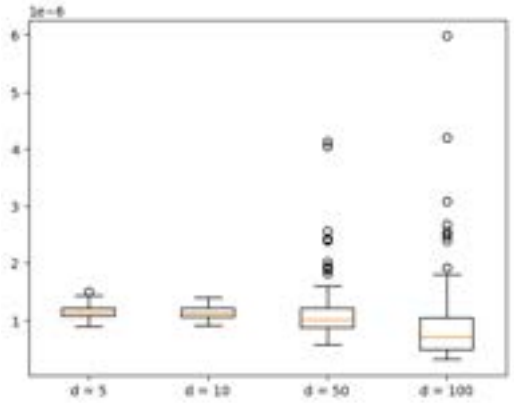


FIGURE 1.5 – Boite à moustache pour les différents estimateurs P_f^{SS} correspondant à chaque dimension d pour le cas $\beta = 5$ et $P_f = 1.15 \times 10^{-4}$.

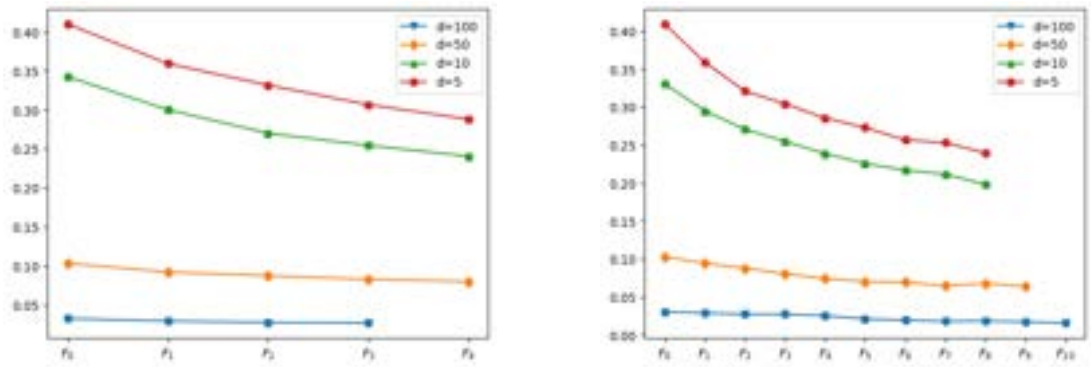
Dans ce cas, l'utilisation de la méthode SS est plus avantageuse en termes de budget d'appel à la fonction Φ (valeur ν plus élevée que pour le cas précédent résumé dans la Table 1.1). Cependant, pour les grandes dimensions, comme 50 et 100, les performances se dégradent considérablement, et l'estimation devient beaucoup moins précise. Les résultats sont donnés en Table 1.2.

Remark 4 *Les taux d'acceptation moyens pour les événements $(F_i)_i$ communs aux deux paramétrages ($\beta = 5$ et $\beta = 3.5$) sont égaux, car on considère la même probabilité fixe p_0.*

Il est important de noter que les entrées pour cet exemple-test sont gaussiennes et indépendantes, ce qui rend le noyau choisi adapter. Cependant, le fléau de la grande dimension empêche une exploration efficace de l'espace d'état de manière vectorielle, car celui-ci devient "*vide*". Nous faisons face à une sparsité des données et il devient difficile de trouver suffisamment de données proches les unes des autres. Comme mentionné précédemment, il existe des variantes des algorithmes MCMC qui permettent de décomposer le problème de grande dimension en plusieurs problèmes unidimensionnels (voir section suivante).

Pour illustrer la remarque 3, on peut constater que les taux d'acceptation moyens diminuent non seulement au fur et à mesure que l'on progresse dans les événements $(F_i)_i$, mais également lorsqu'on augmente la dimension d.

Dans la Figure 1.6, on fixe la perturbation σ à 0.4 et on observe la trajectoire des taux d'acceptation moyens pour chaque dimension d.



(a) $\beta = 3.5 \quad P_f = 9.3 \times 10^{-4}$ **(b)** $\beta = 5 \quad P_f = 1.5 \times 10^{-6}$

FIGURE 1.6 – Évolution du taux d'acceptation moyen pour chaque dimension d en fonction de l'événement F_i considéré. La perturbation σ est la même pour chaque dimension.

V Modified Metropolis Algorithm (MMA)

L'algorithme MMA, introduit dans le papier [1], offre une solution pour surmonter les limitations du noyau gaussien classique des MCMC en grande dimension. Plutôt que de générer des échantillons à partir d'une distribution multidimensionnelle, cet algorithme procède en générant les échantillons coordonnées par coordonnées.

Cependant, une condition nécessaire doit être respectée : le vecteur aléatoire $X = (X_1, \dots, X_d)$ doit avoir des composantes indépendantes, i.e que f_X doit se composer comme un produit de d densités

$$f_X(x) = \prod_{k=1}^{d} f_k(x_k), \quad x = (x_1, \dots, x_d) \in \mathbb{R}^d. \tag{1.12}$$

V.1 Principe du fonctionnement de l'algorithme MMA

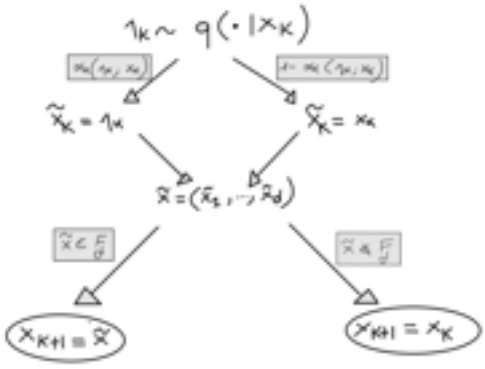
Soit $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, un échantillon issu de la distribution $f_{X|F_j}$ et $(q_k(. \mid x_k))_{k=1}^d$ les noyaux de propositions unidimensionnelles. La génération d'une proposition $\tilde{x}$ se fait par la méthode suivante :

Pour tout $k \in \{1, \dots, d\}$

1. Tirer un échantillon $\eta_k \sim q(. \mid x_k)$
2. Calculer le ratio
$$\alpha_k(\eta_k, x_k) = \frac{f_k(\eta_k) q_k(x_k \mid \eta_k)}{f_k(x_k) q_k(\eta_k \mid x_k)}$$
3. Construction du vecteur de proposition $\xi \in \mathbb{R}^d$:
$$\xi = \begin{cases} \eta_k & \text{avec une probabilité } \alpha_k(\eta_k, x_k) \\ x_k & \text{avec une probabilité } 1 - \alpha_k(\eta_k, x_k) \end{cases}$$
4. Acceptation ou rejet de ξ
$$\tilde{\mathbf{x}} = \begin{cases} \xi & \text{si } \xi \in F_j \\ \mathbf{x}_k & \text{si } \xi \notin F_j \end{cases}$$

On peut démontrer que le noyau de transition renvoyé par l'algorithme MMA vérifie la condition d'irréversibilité, soit la loi invariante est $f_{X|F_j}$.

On peut illustrer cette procédure par le schéma suivant



V.2 Application de l'algorithme MMA à l'exemple-test 4-branches

Application des SS avec l'algorithme MMA pour le cas : $\beta = 3.5 \quad P_f = 9.3 \times 10^{-4}$

Dimension	$\hat{P_f}^{MMA}$	CV	ν	σ
$d = 5$	9.31×10^{-4}	0.043	2.30	0.4
$d = 10$	9.37×10^{-4}	0.043	2.27	0.4
$d = 50$	9.26×10^{-4}	0.047	2.93	0.4
$d = 100$	9.30×10^{-4}	0.047	1.94	0.4

(a) Résumé de la qualité de l'estimateur P_f^{SS}

Évènement	F_0	F_1	F_2	F_3	F_4
$d = 5$	0.63	0.57	0.53	0.49	0.46
$d = 10$	0.64	0.57	0.52	0.47	0.44
$d = 50$	0.63	0.56	0.50	0.46	0.43
$d = 100$	0.63	0.56	0.50	0.46	0.42

(b) Taux d'acceptation ρ pour chaque événement

TABLE 1.3 – Estimation par l'algorithme par avec **MMA** pour l'événement où $\beta = 3.5$ et $P_f = 9.3 \times 10^{-4}$. La probabilité fixe $p_0 = 0.25$. Les noyaux de proposition sont des marches aléatoires gaussiennes. Les longueurs de chaînes sont de taille 6. Pour les taux d'acceptation ρ, ce sont les taux moyennés sur l'ensemble des chaînes pour chaque événement F_i

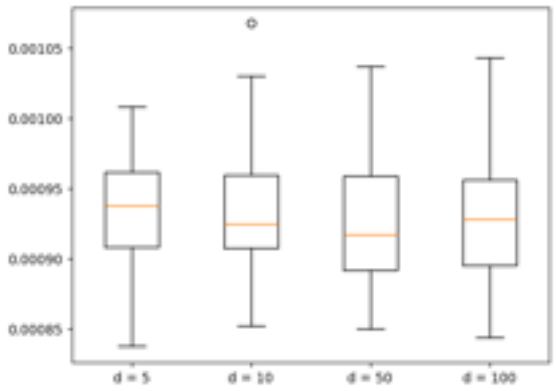


FIGURE 1.7 – Boite à moustache pour les différents estimateurs correspondant à chaque dimension d

Application de l'algorithme MMA pour le cas : $\beta = 5 \quad P_f = 1.15 \times 10^{-6}$

Dimension	$\hat{P_f}^{MMA}$	CV	ν	σ
$d = 5$	1.14×10^{-6}	0.05	620.08	0.4
$d = 10$	1.15×10^{-6}	0.06	468.20	0.4
$d = 50$	1.17×10^{-6}	0.05	648.51	0.4
$d = 100$	1.16×10^{-6}	0.06	470.18	0.4

(a) Résumé de la qualité de l'estimateur P_f^{SS}

Évènement	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
$d = 5$	0.63	0.58	0.52	0.48	0.46	0.44	0.42	0.40	0.38
$d = 10$	0.63	0.56	0.51	0.48	0.44	0.42	0.40	0.38	0.38
$d = 50$	0.63	0.56	0.50	0.46	0.42	0.40	0.37	0.35	0.32
$d = 100$	0.64	0.56	0.50	0.46	0.42	0.39	0.37	0.35	0.33

(b) Taux d'acceptation ρ pour chaque événement

TABLE 1.4 – Estimation par l'algorithme **MMA** pour l'événement où $\beta = 5$ et $P_f = 1.5^{-6}$. La probabilité fixe $p_0 = 0.25$. Les noyaux de proposition sont des marches aléatoires gaussiennes. Les longueurs de chaînes sont de taille 6. Pour les taux d'acceptation ρ, ce sont les taux moyennés sur l'ensemble des chaînes pour chaque événement $(F_i)_i$

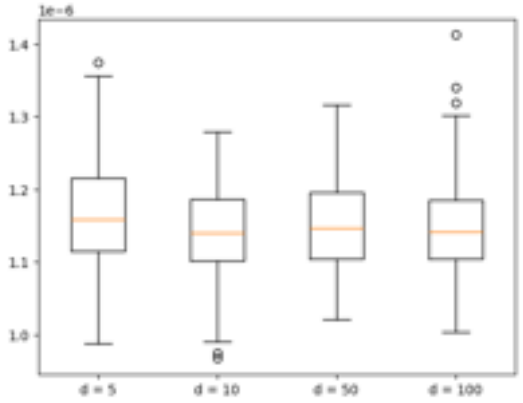


FIGURE 1.8 – Boites à moustache des différents estimateurs de l'exemple $\beta = 5$ et $P_f = 1.5^{-6}$ pour chaque cas de dimension d

L'algorithme **MMA** renvoie des estimations de P_f plus précises avec de faibles coefficients de variation (CV). On observe notamment la robustesse de la méthode pour des événements à très faible occurrence ($\beta = 5$ et $P_f = 1.5 \times 10^{-6}$). En effet, les coefficients ν sont bien plus larges pour ce cas.

Enfin, la dimension n'influence pas sur le taux d'acceptation moyen, contrairement à la méthode SS classique (voir 1.6b et 1.6a).

L'algorithme SS est une solution efficace pour estimer des probabilités de faible occurrence, mais il présente certaines limites. Parmi celles-ci, on observe une diminution du taux d'acceptation au fur et à mesure que l'on progresse dans les événements F_j, ainsi que l'apparition d'un biais lorsque la dimension d augmente. L'élément central de cet algorithme est l'utilisation d'algorithmes MCMC. Dans un cadre idéal, un échantillonnage précis pour chaque événement de la séquence $(F_j)_j$ permettrait d'obtenir des estimations de P_f plus fiables. Pour améliorer ce processus, il est crucial de disposer d'un noyau de proposition Q de qualité.

La remarque 1 nous incite à envisager les densités définies dans (1.8) ou, à défaut, une bonne approximation de celle-ci, pour obtenir des échantillons pertinents. Plusieurs approches sont disponibles dans la littérature pour ce faire :

1. *Distributions paramétriques* : Ces méthodes offrent une bonne robustesse face à la grande dimension, mais leur nombre élevé de paramètres, ainsi que leur manque de flexibilité (formes de distributions définies a priori), en font des candidats moins idéaux.
2. *Distributions non paramétriques* : Contrairement aux distributions paramétriques, ces méthodes sont plus flexibles, mais elles manquent de robustesse face à la grande dimension

Les récents travaux sur les modèles génératifs nous ont amenés à envisager l'utilisation de ces modèles pour l'apprentissage de notre loi cible f_X. Ces modèles ont démontré à la fois de la flexibilité et de la robustesse dans des applications récentes, ce qui en fait une option prometteuse pour surmonter les limitations identifiées dans l'algorithme SS.

Chapitre 2

Auto-encodeurs Variationnel (VAE)

Dans cette section, nous nous concentrons sur une classe spécifique de modèles d'apprentissage : les modèles génératifs. Ces modèles probabilistes sont particulièrement adaptés à l'apprentissage de distributions complexes, telles que les distributions multimodales ou celles définies dans des espaces de grande dimension. Basés sur des réseaux de neurones artificiels [9], ils sont largement utilisés dans la génération d'images, de vidéos et de textes [16].

Les modèles génératifs peuvent être classés en trois grandes catégories :

1. Modèles autorégressifs (AR)
2. Modèles basés sur les transformations de flux (flow-based models)
3. Modèles à variables latentes.

Dans cette étude, nous nous intéressons plus particulièrement aux méthodes impliquant l'utilisation de variables latentes. L'idée derrière ces méthodes est de supposer l'existence d'un espace de plus faible dimension $\mathcal{Z}$, appelé espace latent, puis de procéder à la méthode de génération suivante.

$$\begin{cases} z \sim p, & z \in \mathcal{Z} \\ x \sim p(. \mid z), & x \in \mathcal{X}. \end{cases}$$

En résumé, la distribution des les variables latentes z correspond aux informations cachées dans les données et les distributions conditionnelles $p(. \mid z) \quad \forall z \in \mathcal{Z}$ peuvent être vu comme des générateurs.

Le premier enjeu d'une telle méthode serait alors de définir l'espace latent. Étant donné que le but de l'espace latent est de résumer nos données, on peut considérer les méthodes de réduction de dimension.

I Auto-encodeurs : une Méthode de réduction de dimension

Nous résumons les données sous la forme d'une matrice $\mathbb{X} \in \mathbb{R}^{n \times d}$, qui représente une collection de n vecteurs de dimension d.

I.1 Principes et techniques de la réduction de dimension

La réduction de dimension consiste, dans la plupart des cas, à trouver des matrices de plus petite taille, c'est-à-dire avec un nombre réduit de lignes n ou de colonnes d, tout en préservant une certaine similarité avec la matrice initiale $\mathbb{X}$.

Un exemple préliminaire de réduction de dimension est la décomposition UV. L'objectif est d'approcher $\mathbb{X}$ par un produit de matrices :

$$\mathbb{X} = U^t V,$$

où

$$\mathbb{X} = {}^t\left[{}^tX^{(1)}, \ldots, {}^tX^{(n)} \right].$$

Supposons que nous souhaitons projeter chaque donnée $X^{(i)} \in \mathbb{R}^d$ pour tout $i \in \{1, \ldots, n\}$ dans un espace de dimension réduite $\mathcal{Z}$, par exemple $\mathbb{R}^k$ avec $k < d$. Nous considérons d'abord une représentation des données dans $\mathcal{Z}$:

$$U = {}^t\left[{}^tU^{(1)}, \ldots, {}^tU^{(n)} \right] \in \mathbb{R}^{n \times k.}$$

De manière similaire, nous définissons la matrice

$$V = \left[{}^tV^{(1)}, \ldots, {}^tV^{(n)} \right] \in \mathbb{R}^{d \times k},$$

qui constitue la base des vecteurs pour la nouvelle représentation des données.

Le problème d'optimisation peut alors être formulé comme suit :

$$\min_{(U,V)} || \mathbb{X} - U^tV ||^2_{frob}. \tag{2.1}$$

où $|\cdot|_{Frob}$ représente la norme de Frobenius.

Des contraintes peuvent être imposées sur U et V. Par exemple, en imposant une contrainte d'orthogonalité, on retrouve la méthode Singular Value Decomposition (SVD). C'est une technique fondamentale en algèbre linéaire qui décompose une matrice en trois autres matrices, révélant des propriétés essentielles de la matrice d'origine

$$\mathbb{X} = U\Sigma^tV.$$

— U est une matrice orthogonale de dimension $n \times n$
— Σ est une matrice diagonale de dimension $n \times d$
— V est une matrice orthogonale de dimension $d \times d$

En sélectionnant les premières valeurs singulières et les vecteurs associés (troncature de la SVD), on peut obtenir une approximation de la matrice $\mathbb{X}$ avec une dimension réduite.

On peut quantifier l'erreur qu'on commet dans notre approximation.
On suppose que $rg(\mathbb{X}) = r$ (en notant rg le rang d'une matrice). alors pour $B \in \mathbb{R}^{n \times d}$ et $q = \min\{r, rg(B)\}$, $q < r$ on a :

$$\min_{B, rg(B) \leq q} || \mathbb{X} - B ||^2_{frob} = \sum_{k=q+1}^{r} \sigma(\mathbb{X})^2.$$

où $\sigma(\mathbb{X})^2$ sont les valeurs singulières de la matrice $\mathbb{X}$.

Une application spécifique de SVD aux données centrées est l'Analyse en Composantes Principales (ACP), une technique de réduction de dimensionnalité qui transforme les données en un nouvel ensemble de variables non corrélées appelées composantes principales. Ces composantes sont ordonnées de manière à ce que la première retienne le plus de variance possible, la deuxième retient le plus de variance possible sous contrainte d'être orthogonale à la première, et ainsi de suite.

Or la méthode de l'ACP présente des limites :

— **Linéarité** : L'ACP repose sur des transformations linéaires et peut échouer à capturer les relations non linéaires complexes dans les données.
— **Scalabilité** : Pour des ensembles de données très larges, l'ACP peut devenir coûteuse en termes de calcul, ce qui a conduit au développement de l'ACP randomisée [14].

Il s'agit donc de développer une méthode qui offre une réduction de dimension plus flexible, capable de capturer des motifs complexes au-delà de la simple linéarité, tout en maintenant un budget de calcul raisonnable.

I.2 Les Auto-Encodeurs

Les Auto-Encodeur (AE) sont une classe de réseaux de neurones non supervisés utilisée pour apprendre une représentation compacte des données d'entrée. Le but est d'obtenir une solution au

problème d'optimisation (2.1) à l'aide d'une architecture neuronale.
Un AE se compose de trois parties principales :

— Un **Encodeur** noté E_ϕ. C'est un réseau de neurones paramétré par les poids $\phi \in \Xi$
— Un espace latent $\mathcal{Z}$
— Un **Décodeur**, noté D_θ. C'est un réseau de neurones paramétré par les poids $\theta \in \Theta$

Dans un premier temps, l'encodeur comprime l'observation $x \in \mathbb{R}^d$ et lui attribut une représentation $z = E_\phi(x) \in \mathcal{Z}$. Dans un second temps, le décodeur réalise la procédure inverse en essayant de reconstruire le point initial x à partir de l'encodage z et on a $D_\theta(z) = \hat{x} \in \mathbb{R}^d$, voir la Figure 2.1.

L'objectif d'un AE est de minimiser la différence entre l'entrée x et sa reconstruction $\hat{x}$. Cette différence est quantifiée par une fonction de perte, typiquement une fonction de type erreur quadratique moyenne (Mean Squared Error, MSE) :

$$Obj(x, \hat{x}) = || \, x - \hat{x} \, ||^2 = || \, x - D_\theta(E_\phi(x)) \, ||^2, \tag{2.2}$$

où $|| \, . \, ||^2$ représente la norme L_2.

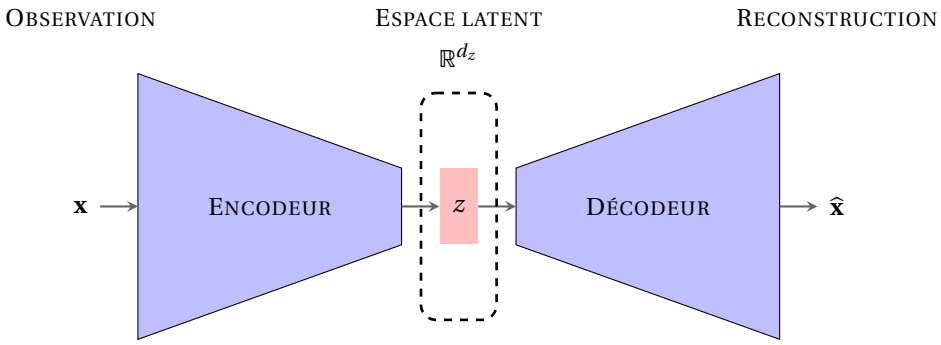


FIGURE 2.1 – Principe du fonctionnement d'un auto-encodeur

Cependant, les auto-encodeurs (AE) rencontrent des difficultés en matière de généralisation ; leur capacité à générer de nouvelles données similaires à celles utilisées pour l'entraînement est limitée.

Remark 5 Le principe de fonctionnement de l'AE peut être reformulé de manière à correspondre aux formulations présentées au début de cette section I.1.
Considérons les transformations linéaires suivantes :

$$W : \mathbb{R}^d \to \mathbb{R}^k \qquad V : \mathbb{R}^k \to \mathbb{R}^d$$

où ${}^tW$ et ${}^tV$ représentent respectivement les poids des réseaux de neurones E_ϕ (encodeur) et D_θ (décodeur) avec des biais nuls. En utilisant l'identité comme fonction d'activation, chaque ligne de $\mathbb{X}$, $X^{(i)}$, est transformée en $z_i = WX^{(i)}$, ce qui définit la matrice U mentionnée précédemment :

$$U = {}^t\left[\, {}^t z^{(1)}, \ldots, {}^t z^{(n)} \right].$$

En sortie de l'auto-encodeur (AE), chaque donnée $X^{(i)}$ est reconstruite par la transformation $VWX^{(i)} = Vz_i$. Par transposition, les valeurs reconstruites $\hat{X}$ sont données par U^tV. Pour que $\mathbb{X} \approx U^tV$ soit satisfait, il est nécessaire de résoudre le problème d'optimisation suivant :

$$\max_{(\phi,\theta)} || \, \mathbb{X} - U^tV \, ||^2_{frob}$$

ce qui est similaire à l'équation (2.1).

II Fonctionnement d'un VAE

II.1 Généralités sur les VAE

Les VAE [7] constituent une classe de modèles génératifs qui étendent le principe des auto-encodeurs classiques. En effet, un VAE se compose également d'un encodeur E_ϕ et d'un décodeur D_θ, mais à la différence des auto-encodeurs classiques, les deux composants génèrent des distributions de probabilité plutôt que des sorties déterministes.

L'objectif principal d'un VAE est d'approximer une distribution cible, notée ici f_X. Pour ce faire, une variable latente (non observable) Z est introduite, permettant d'établir la relation suivante :

$$f_X(x) \approx f_\theta(x) = \int p_\theta(x \mid z)\ p(z)dz \quad \forall x \in \mathbb{R}^d \tag{2.3}$$

Cette équation représente une intégration sur l'espace des variables latentes $\mathcal{Z}$. On peut interpréter cette représentation comme un problème d'inférence bayésienne :

1. p est la loi de Z, appelée la loi a priori.
2. $p_\theta(. \mid z)$ est la vraisemblance.
3. $p_\theta(. \mid x)$ est la loi a posteriori.

Dans le cadre classique, on suppose que $p_\theta(. \mid z)$ suit une distribution gaussienne $\mathcal{N}(\mu_z^\theta, \Sigma_z^\theta)$ paramétrée par le décodeur D_θ, avec p comme distribution continue. Ainsi, le VAE approxime f_X par un mélange infini de gaussiennes.

L'intégrale (2.3) étant souvent difficile à calculer, les VAE adoptent une approche variationnelle en introduisant une distribution approximative $q_\phi(. \mid x)$, paramétrée par l'encodeur E_ϕ et appelée distribution a posteriori variationnelle. Cette distribution est sélectionnée au sein d'une famille paramétrique $\mathcal{P}$, permettant une génération et une évaluation de la densité relativement simples. Dans le cadre de notre étude, cherchant à approximer un vecteur aléatoire à densité continue, nous choisissons $q_\phi(. \mid x) \overset{\mathcal{L}oi}{=} \mathcal{N}(\mu_x^\phi, \Sigma_x^\phi)$.

La procédure d'encodage et de reconstruction d'un point $x \in \mathbb{R}^d$ par VAE est représente dans la Figure 2.2 et donné par le schéma qui suit :

1. L'encodeur E_ϕ renvoie pour chaque donnée x les paramètres de la distribution a posteriori variationnel $q_\phi(. \mid x) : (\mu_x^\phi, \Sigma_x^\phi) = E_\phi(x)$
2. Génération d'un point $z \sim q_\phi(. \mid x)$ dans l'espace latent
3. Le décodeur renvoi pour chaque donnée z les paramètres de la distribution a posteriori $p_\theta(. \mid z) : (\mu_z^\theta, \Sigma_z^\theta) = D_\theta(z)$
4. Génération de la reconstruction $\hat{x} \sim p_\theta(. \mid z)$

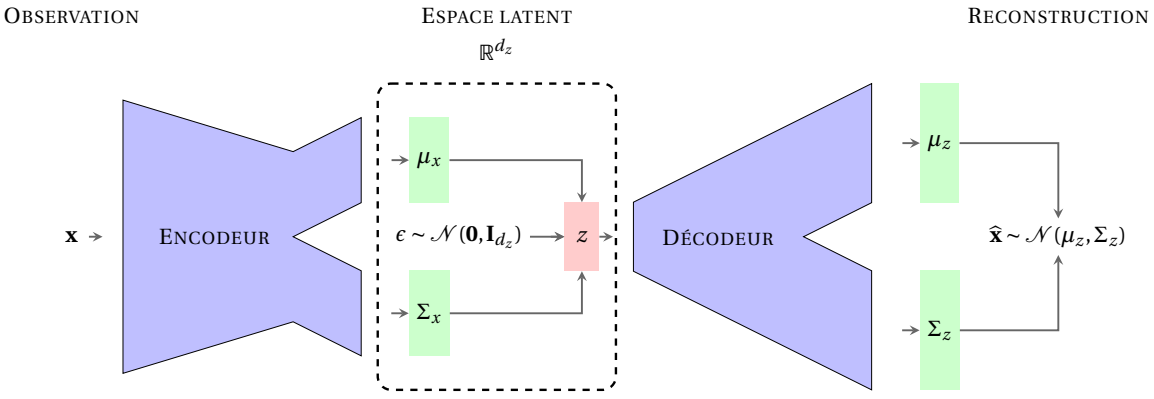


FIGURE 2.2 – Principe du fonctionnement d'un auto-encodeur variationnel utilisant une distribution standard gausienne fixe

Une fois le principe de fonctionnement du VAE établit, on procède à l'entraînement des poids ϕ et θ de E_ϕ et D_θ de manière à ce que f_θ soit une bonne approximation de la densité cible f_X.

II.2 Entraînement d'un VAE

Pour l'entraînement, on s'intéresse à la log-vraisemblance de f_θ et plus précisément, on cherche à maximiser par rapport à θ sous f_X :

$$\max_{\theta\in\Theta} \quad \mathbb{E}_{f_X}[log(f_\theta)] \tag{2.4}$$

On obtient alors le développement qui suit

$$\begin{aligned}
log(f_\theta) &= log\left(\int p_\theta(x \mid z)\ p(z)dz\right) && \text{avec équation (2.3)}\\
&= log\left(\int \frac{p_\theta(x \mid z)\ p(z)}{q_\phi(z \mid x)} q_\phi(z \mid x)dz\right)\\
&\geq \int log\left(\frac{p_\theta(x \mid z)\ p(z)}{q_\phi(z \mid x)}\right) q_\phi(z \mid x)dz && \text{par inégalité de Jensen}\\
&= \mathbb{E}_{q_\phi(.|x)}\left[log\left(\frac{p_\theta(x \mid Z)\ p(Z)}{q_\phi(Z \mid x)}\right)\right]\\
&= \mathbb{E}_{q_\phi(.|x)}[p_\theta(x \mid Z)] - \mathbb{E}_{q_\phi(.|x)}\left[log\left(\frac{p(Z)}{q_\phi(Z \mid x)}\right)\right]\\
&= \mathbb{E}_{q_\phi(.|x)}[p_\theta(x \mid Z)] - D_{KL}(q_\phi(. \mid x) \mid\mid p) && \text{par définition de la divergence de Kullback-Leibler}
\end{aligned}$$

Par linéarité de l'espérance, on a :

$$\mathbb{E}_{f_X}[log(f_\theta(X))] \geq \mathbb{E}_{f_X}[\mathbb{E}_{q_\phi(.|x)}[p_\theta(X \mid z)]] - \mathbb{E}_{f_X}[D_{KL}(q_\phi(. \mid x)] \tag{2.5}$$

Maximiser la borne inférieure équivaut à résoudre le problème d'optimisation décrit par (2.4). Cette borne est connue sous le nom de Evidence Lower BOund (ELBO).

$$\mathscr{L}(\theta,\phi) = \mathbb{E}_{f_X}[\mathbb{E}_{q_\phi(.|x)}[p_\theta(X \mid Z)]] - \mathbb{E}_{f_X}[D_{KL}(q_\phi(. \mid x) \mid\mid p)] \tag{2.6}$$

On peut définir la valeur de l'ELBO pour une réalisation $x \in \mathbb{R}^d$ de X comme suit :

$$L(x,\theta,\phi) = \underbrace{\mathbb{E}_{q_\phi(.|x)}[log(p(Z \mid x))]}_{\text{reconstruction}} - \underbrace{D_{KL}(q_\phi(. \mid x) \mid\mid p)}_{\text{régularisation}} \tag{2.7}$$

Elle est composée de deux termes qui ont un comportement opposé lors de l'optimisation de (2.4). Maximiser ELBO revient à trouver un compromis entre ces deux termes.
Maximiser le terme de régularisation revient à optimiser les poids (θ,ϕ) de telle sorte que les distributions $q_\phi(. \mid x)$ aient des supports disjoints au sein de l'espace latent. Ceci permettra au décodeur de reconstruire fidèlement x à partir de la distribution encodée associée.
Quant au terme de régularisation, il régit la similarité entre le prior p et le posterior variationnel $q_\phi(. \mid x)$. L'objectif est que toutes les distributions $q_\phi(. \mid x)$ soient proches du prior.

Si on s'intéresse de plus près à ce terme, on a la décomposition suivante :

$$-D_{KL}(q_\phi(. \mid x) \mid\mid p) = \underbrace{-\mathbb{E}_{q_\phi(.|x)}[log(q_\phi(. \mid x))]}_{\text{Entropie}} - \underbrace{(-\mathbb{E}_{q_\phi(.|x)}[log(p(Z))])}_{\text{entropie croisée}} \tag{2.8}$$

Sous les hypothèses gaussiennes, on peut aisément calculer la valeur de l'entropie

$$\text{Entropie} = \frac{1}{2}\sum_{i=1}^{d} log[2e\pi(\Sigma_x^\phi)_{ii}] \tag{2.9}$$

Étant donné qu'on cherche à maximiser (2.9), il faut $\sigma_x^\phi(i)^2 \to +\infty$. L'entropie cherche à étaler E_ϕ

autant que possible, cependant, ceci n'est pas réalisable en pratique, car le terme de reconstruction force l'encodeur à être centré autour de chaque point.

Pour l'entropie croisée, on a :

$$\text{entropie croisée} = \int_{\mathcal{Z}} q_\phi(z \mid x) log(p(z)) dz = \frac{d}{2} log(2\pi) + \frac{1}{2}\left((\Sigma_x^\phi)_{ii} + \mu_x^\phi(i)^2\right) \tag{2.10}$$

Lorsque nous cherchons à maximiser l'ELBO, cela implique de maximiser l'entropie croisée, c'est-à-dire d'essayer d'aligner le prior p avec le postérieur variationnel $q_\phi(. \mid x)$. Maintenant, examinons le comportement de l'entropie croisée lorsque l'on impose un prior gaussien standard $\mathcal{N}(0, I_d)$. L'objectif est de faire correspondre cette forme fixe au fur et à mesure de l'entraînement. Cependant, le problème est que le décodeur, en se basant sur le terme de reconstruction, aura tendance à générer des distributions centrées autour des points d'entraînement, ce qui peut entraîner l'apparition de "zones blanches". Dans ces zones, $q_\phi(. \mid x)$ attribuera une très faible probabilité, tandis que le prior les considérera comme des régions à forte masse de probabilité.

Ce décalage deviendra apparent lors du processus de génération, comme expliqué au début de la Section 2. Lorsque des points issus de ces zones blanches seront générés à partir du prior, le décodeur, n'ayant que peu ou pas appris sur ces zones, produira des échantillons de qualité inférieure. Dans ce cas, il peut être plus judicieux de considérer des priors non fixes et dont les paramètres sont mis à jour au fur à mesure de l'entraînement.

II.3 Choix de prior flexible

Classiquement, la distribution normale standard est le priori utilisé dans plusieurs applications du VAE mais comme expliquée, il est possible de choisir un autre type de prior :

$$p_\lambda(z) \qquad \lambda \in \Lambda \text{ une famille de paramètres}$$

Une possibilité d'a priori flexible peut être un mélange gaussien :

$$p_\lambda(z) = p_\lambda^{MoG}(z) = \sum_{l=1}^{K} w_l g_{(m_l, s_l)}(z) \qquad \lambda = \{w_l, m_l, s_l\}_{l=1}^K \tag{2.11}$$

Où, $g_{(m_l,s_l)}$ est la densité d'une gaussienne de moyenne m_l et d'écart-type s_l.

Ce prior Mélange de gaussiennes (MoG) permet une plus grande flexibilité dans la modélisation des données en capturant des structures plus complexes, car il représente la distribution latente comme un mélange de plusieurs gaussiennes. En plus de générer les échantillons latents, le VAE apprend à estimer les paramètres des différentes composantes du mélange, améliorant ainsi la capacité du modèle à capturer des distributions multimodales et à générer des échantillons plus diversifiés. L'expression du ELBO change et on a la formulation qui suit :

$$\begin{aligned}\mathcal{L}(\theta, \phi, \lambda) &= \mathbb{E}_{f_X}[\mathbb{E}_{q_\phi(.|x)}[p_\theta(X \mid Z)]] - \mathbb{E}_{f_X}[D_{KL}(q_\phi(. \mid x) \mid\mid p_\lambda)] \\ &= \mathbb{E}_{f_X}[\mathbb{E}_{q_\phi(.|x)}[p_\theta(X \mid Z)]] + \mathbb{E}_{f_X}[-\mathbb{E}_{q_\phi(.|x)}[log(p_\phi(X \mid Z))]] + \mathbb{E}_{f_X}[\mathbb{E}_{q_\phi(.|x)}[log(p_\lambda)]]\end{aligned} \tag{2.12}$$

Avec

$$\mathbb{E}_{f_X}[\mathbb{E}_{q_\phi(.|x)}[log(p_\lambda^{MoG})(Z)]] = \int_{\mathcal{X}} \int_{\mathcal{Z}} log\left[\sum_{l=1}^{K} w_l g_{(m_l,s_l)}(z)\right] q_\phi(z \mid x) dz dx \tag{2.13}$$

L'équation (2.13) étant intractable, on approxime celle-ci par une méthode de Monte-Carlo lors de l'entraînement. On s'appuie sur l'échantillon $z = \mu_x^\phi + (\Sigma_x^\phi)^{1/2}\epsilon$ généré par la procédure de reparamétrisation à la sortie de l'encodeur E_ϕ. On a alors :

$$\mathbb{E}_{f_X}[-\mathbb{E}_{q_\phi(.|x)}[log(p_\lambda)(Z)]] \approx \frac{1}{N}\sum_{i=1}^{N} -log[p_\lambda^{MoG}(Z^{(i)})] \qquad Z^{(i)} \sim q_\phi(. \mid X^{(i)}) \tag{2.14}$$

Avec $\left\{X^{(1)},\dots,X^{(N)}\right\}$, l'échantillon d'entraînement.

La quantité $\mathbb{E}_{q_\phi(.|x)}[log(p_\lambda)]$ est évaluée sur un échantillon de taille 1. Ce choix peut paraître surprenant à première vue, mais est en réalité suffisant parce que les estimations associées à chaque observation sont en pratique moyennées par mini-batch. Un schéma récapitulatif du fonctionnement de ce VAE est présenté dans la Figure 2.3.

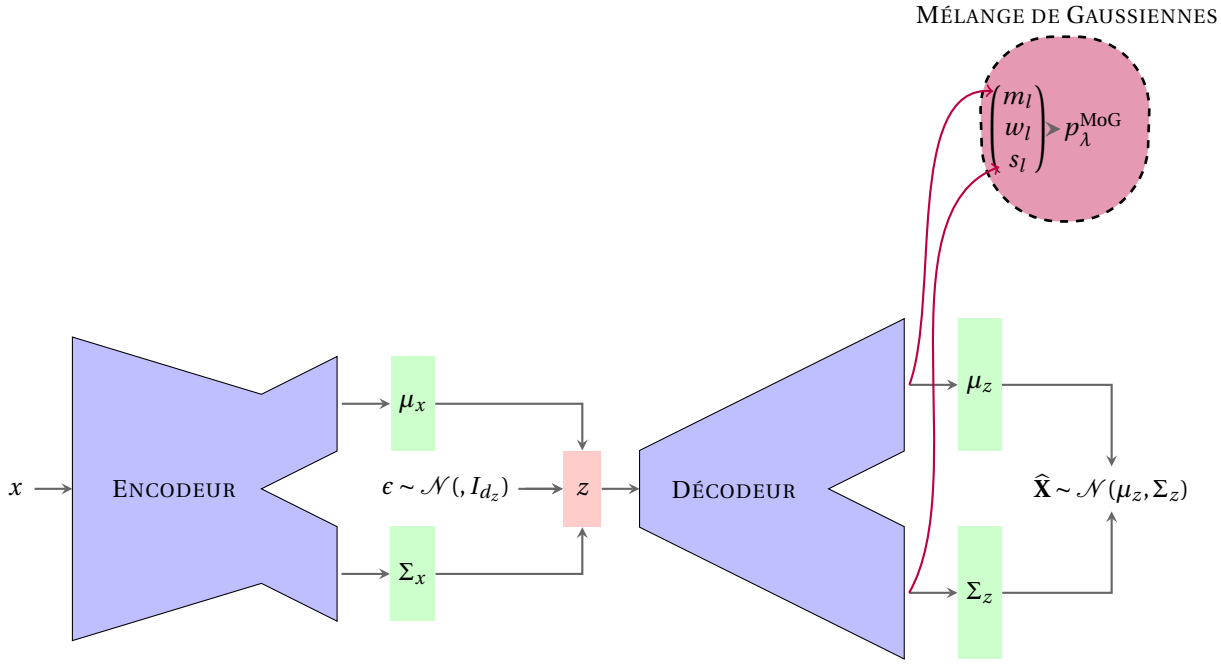


FIGURE 2.3 – Principe de fonctionnement d'un VAE utilisant un prior MoG : contrairement à un VAE avec un prior gaussien classique, ici, le modèle récupère non seulement les paramètres des distributions de E_ϕ et D_θ, mais également ceux du mélange.

La deuxième possibilité considérée dans notre étude est le VAE utilisant le prior VampPrior (VP) développé dans l'article [15]. L'idée est de se baser sur les travaux dans l'article [8]. En cherchant le prior optimal dans le cadre des VAE, la réécriture dans (2.12), permet de voir l'apparition de l'entropie croisée qui fait intervenir le prior p_λ. On peut donner la définition suivante

Definition 8 (Distribution a posteriori agrégée) La distribution a posterior agrégée associée à un VAE dont les paramètres de l'encodeur E_ϕ et le décodeur D_θ sont définis par :

$$q_\phi(z) = \int_{\mathcal{X}} q_\phi(z \mid x) f_X(x)\, dx = \mathbb{E}_{X\sim f_X}[q_\phi(z \mid X)] \qquad \forall z \in \mathcal{Z} \tag{2.15}$$

La distribution a posteriori agrégée peut être vue comme la mixture infinie des distributions a posteriori variationnelle pondérée par la distribution des observations f_X. On réécrit l'ELBO en réexprimant le dernier terme d'entropie croisée avec le posteriori agrégé p_ϕ :

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{f_X}[\mathbb{E}_{q_\phi(.|x)}[p_\theta(X \mid Z)]] + \mathbb{E}_{f_X}[-\mathbb{E}_{q_\phi(.|x)}[log(p_\phi(X \mid Z))]] + \mathbb{E}_{p_\phi}[log(p_\lambda(Z))]. \tag{2.16}$$

Comme énoncé pour le MoG, maximiser l'ELBO équivaut à maximiser l'entropie croisée. Ceci donne lieu à l'assertion suivante (énoncé établi dans [8])

$$q_\phi = \arg\max_{p_\lambda \in \mathbb{L}^2} \mathbb{E}\Big[log(p_\lambda(Z))\Big]. \tag{2.17}$$

Dans le cadre empirique, nous avons l'expression optimale suivante :

$$p^{emp}_{\phi,opt} = \frac{1}{N}\sum_{i=1}^{N} p_\phi(z \mid X^{(i)}) \qquad \forall z \in \mathcal{Z} \tag{2.18}$$

Une limitation de ce prior est que lorsque le nombre de composantes N devient grand, le calcul des valeurs de cette densité devient informatiquement coûteux. En outre, l'utilisation de ce prior peut également augmenter le risque de sur-apprentissage, comme l'ont souligné [5] et [8].

La solution proposée dans [15] consiste à approcher la distribution a posteriori agrégée par un mélange de postérieurs variationnels établis à partir de *pseudo-inputs*, ce qui constitue le prior VP. Ce prior est défini par l'équation suivante :

$$p_\lambda(z) = \frac{1}{K}\sum_{k=1}^{K} q_\phi(z \mid u_k). \tag{2.19}$$

où $\lambda = \{\phi, u_1, \ldots, u_K\}$ et $\{u_1, \ldots, u_K\} \in (\mathcal{X})^K$ représentent les *pseudo-inputs*. Ces vecteurs sont mis à jour au fur et à mesure de l'apprentissage. En pratique, un réseau de neurones VP_λ, $\lambda \in \Lambda$ est utilisé pour définir ces pseudo-inputs. Le réseau VP_λ prend en entrée K vecteurs $(e_k)_{k=1}^K$, qui forment la base canonique de $\mathbb{R}^k$ et renvoie les *pseudo-inputs* $(u_k)_{k=1}^K$, où chaque $x_k \in \mathcal{X}$.

Un schéma explicatif du VAE avec prior VP est présenté dans la Figure 2.4.

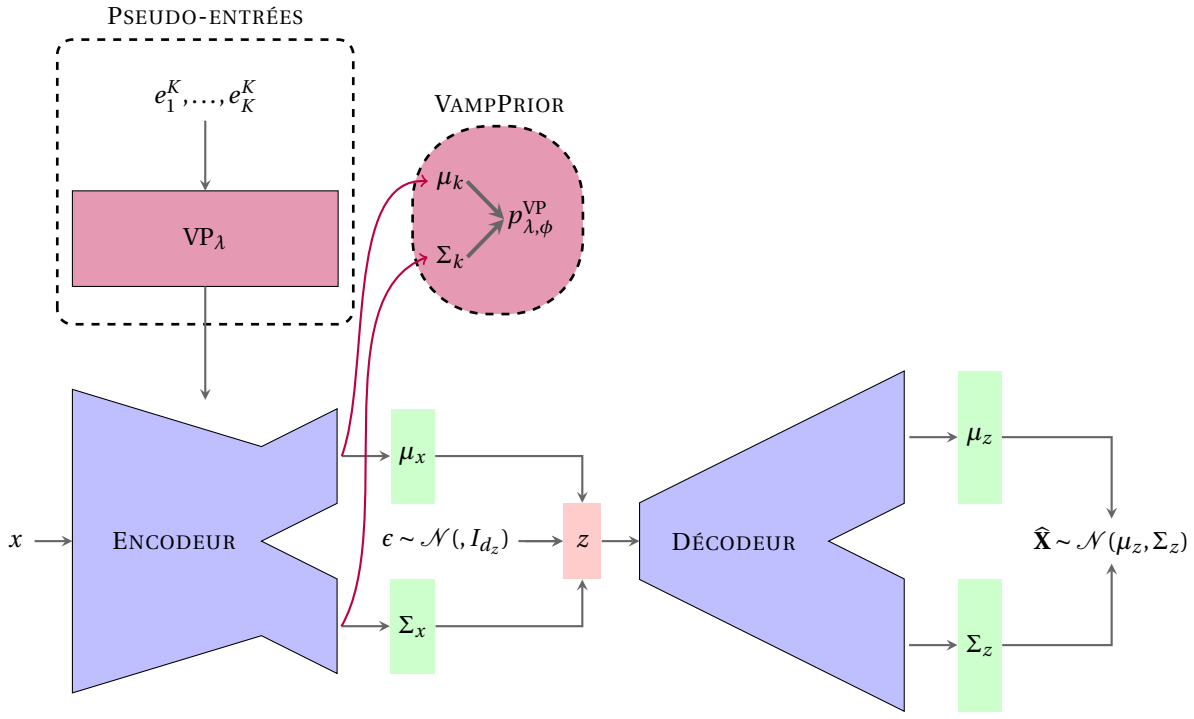


FIGURE 2.4 – Principe de fonctionnement du VAE avec prior VP : Contrairement au modèle classique, ici, le prior $p_{\phi,u_1,\ldots,u_K}$ est un mélange de distributions dont les paramètres sont déterminés non seulement par l'encodeur E_ϕ mais également par un réseau de neurones VP_λ. Ce réseau génère les *pseudo-inputs* qui sont utilisés pour modéliser la distribution a priori.

Maintenant que nous avons discuté des différents modèles de VAE qu'on considérera dans le cadre de ce stage, il est important d'aborder les problématiques que ces modèles rencontrent lors de l'entraînement.

II.4 Procédure d'initialisation

Lors de l'entraînement, la maximisation de l'ELBO implique un équilibre délicat entre les deux termes qui la composent : le terme de régularisation et celui de reconstruction. Cependant, il arrive fréquemment que l'on observe un effondrement de la distribution a posteriori, un phénomène bien connu sous le nom de *posterior collapse*, tel qu'examiné dans la littérature [2]. Ce problème survient lorsqu'une sur-régularisation du modèle fait que l'effet du terme de régularisation devient prédominant, au point où la divergence de Kullback-Leibler devient quasi-nulle

$$D_{KL}(q_\phi(. \mid x) \,||\, p_\lambda) \approx 0.$$

Une solution couramment adoptée pour remédier à ce problème est l'utilisation du β-VAE. Cette méthode consiste à introduire un facteur multiplicatif $\beta \in [0,1]$ devant le terme de régularisation $D_{KL}(q_\phi(. \mid x) \,||\, p_\lambda)$, afin de contrôler l'influence de ce dernier sur l'ELBO. Toutefois, cette approche a ses limites, car la nouvelle formulation de l'ELBO ne correspond plus nécessairement à la minimisation de la log-vraisemblance.

Dans notre étude, nous nous appuyons sur la procédure de pré-entraînement développée dans [4]. En effet, l'une des hypothèses les plus couramment avancées pour expliquer le *posterior collapse* est que le modèle reste bloqué dans un maximum local de la fonction ELBO. Pour éviter cela, l'entraînement débute avec des points de départ bien choisis $(\lambda^{(0)}, \phi^{(0)}, \theta^{(0)})$.

Initialisation de ϕ et θ

Les paramètres ϕ et θ correspondent au poids de l'encodeur E_ϕ et D_θ. Pour l'initialisation, on se basera sur le principe du AE :

$$(\phi^{(0)}, \theta^{(0)}) = \underset{\phi,\theta \in \Xi \times \Theta}{\operatorname{argmin}} \mathbb{E}_{f_X}\left[\,|| \, X - D_\theta^\mu(E_\phi^\mu(X)) \,||_2^2 \,\right] \tag{2.20}$$

où $(D_\theta^\mu,\ E_\phi^\mu)$ représentent respectivement le décodeur et l'encodeur, ne prenant en compte que la moyenne en sortie (la rétropropagation ne s'effectue que pour l'entraînement des moyennes).

Initialisation de λ

Pour les paramètres des priors non fixes introduit dans la sous-section précédente, la procédure d'initialisation dépend du type de prior utilisé.

Pour le VAE MoG, on utilise l'encodage des points renvoyés lors l'initialisation de ϕ et θ dans l'espace latent $\mathcal{Z}$ obtenu, puis on procède par algorithme EM pour l'initialisation des paramètres du mélange :

$$\lambda^{(0)} = \left(m_l^{(0)}, w_l^{(0)}, s_l^{(0)}\right)_{l=1}^K = \mathrm{EM}\left[E_{\phi^{(0)}}(X^{(1)}), \ldots, E_{\phi^{(0)}}(X^{(N)})\right] \tag{2.21}$$

Pour le VAE VP, il s'agit de pré-entraîner le réseau de neurones VP_λ de telle sorte à ce que les *pseudo-inputs* soient représentatifs de la distribution cible afin que le la distribution a priori capte ses caractéristiques. Pour ce faire, on tire uniformément et sans remise $\{i_1 \ldots i_K\}$ entiers dans $[\![1, N]\!]$ afin de se créer le sous-échantillon d'observation $\left\{X^{(i_1)}, \ldots, X^{(i_K)}\right\}$ distribués sous f_X. Puis, on cherche $\lambda^{(0)}$ solution de :

$$\underset{\lambda \in \Lambda}{\operatorname{argmin}} \sum_{k=1}^{K} || \, VP_\lambda(e_k) - X^{(i_k)} \,||_2^2 \tag{2.22}$$

II.5 Test des VAE sur un exemple

Afin de s'assurer du fonctionnement du modèle de VAE proposé, on décide de tester celui-ci en apprenant la distribution suivante :

$$f_X(x) = \frac{\varphi(x_1)\mathbf{1}_{|x_1|>2}}{2(1-F(2))} \times \frac{\varphi(x_2)\mathbf{1}_{|x_2|>2}}{2(1-F(2))} \times \prod_{i=3}^{d} \varphi(x_i)\,; \quad x = (x_1, \ldots, x_d) \in \mathbb{R}^d \tag{2.23}$$

Où φ est la densité d'une gaussienne standard $\mathcal{N}(0,1)$ définie sur $\mathbb{R}$.
Il s'agit de la densité d'un vecteur gaussien standard tronqué sur les deux premières dimensions.

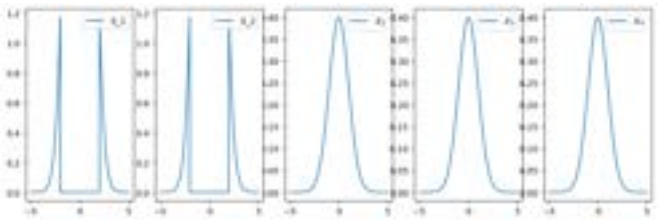


FIGURE 2.5 – Tracé des marginales de la densité-exemple pour la dimension $d = 5$. Les troncatures ne sont faites que sur les deux premières dimensions. On obtient une distribution avec quatre modes importants, soit quatre zones de défaillance.

Nous obtenons une distribution avec quatre modes, et dont la région de défaillance

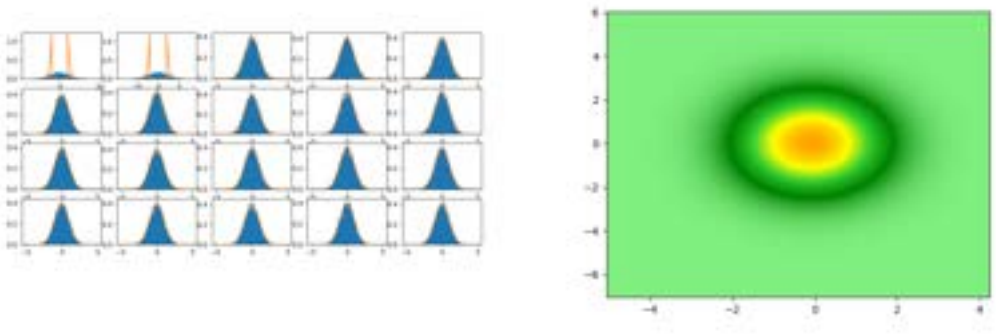
$$D_f = \left\{ x \in \mathbb{R}^d, |x_1| > 2 \text{ et } |x_2| > 2 \right\}$$

se décomposent en quatre zones non connexes.

Dans la suite, on considérera des échantillons issus de la distribution définie par (2.23) pour la dimension $d = 50$. Pour simplifier l'affichage des résultats, on tracera seulement les 20 premières dimensions. Ceci n'enfreindra pas à la transparence des résultats puisque les modes sont présents sur les deux premières dimensions.

II.6 Apprentissage à l'aide du VAE Vamprior

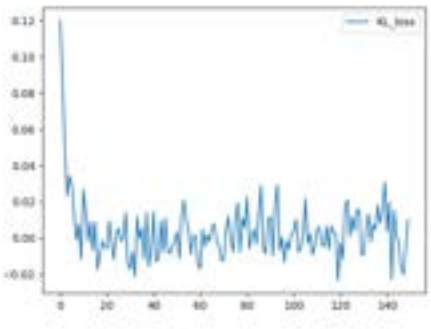
On procède à l'apprentissage de f_X par un VAE muni du prior VP. On utilise une distribution définie sur $\mathbb{R}^{50}$. Dans un premier temps, on illustre l'importance de la procédure d'initialisation. On utilisera un prior muni de 35 pseudo-inputs pour ce cas de figure.



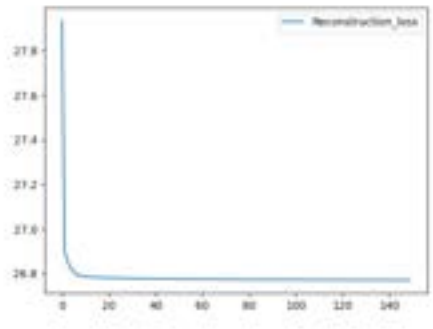
(a) Histogrammes des données générées par VAE-VP. **(b)** Densité du VP sur l'espace latent.

FIGURE 2.6 – Caractéristique de l'apprentissage par VAE-VP sans la procédure de pré-entraînement

Le modèle VAE échoue à repérer les quatre zones. L'espace latent illustre seulement une seule zone compacte. Si on s'intéresse à la trajectoire des pertes D_{KL} et de reconstruction, on remarquera que la première perte n'est pas stable tandis que la deuxième stagne très tôt dans l'apprentissage. En somme, la trajectoire de l'ELBO n'est pas stable au cours de l'apprentissage et surtout, elle oscille. Les résultats sont illustrés dans les Figures 2.7a et 2.7b.

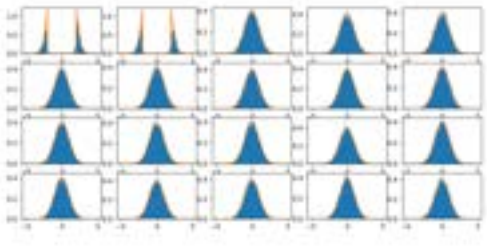


(a) Trajectoire de la perte de Kullback-Leibler lors de l'apprentissage du VAE-VP.

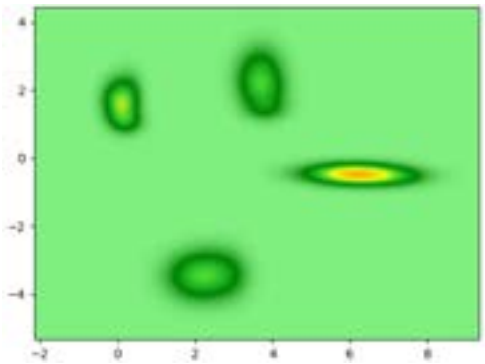


(b) Trajectoire de la perte de reconstruction lors de l'apprentissage du VAE-VP

A la différence, lorsqu'on applique la procédure de pré-entraînement, nous avons les résultats suivants :



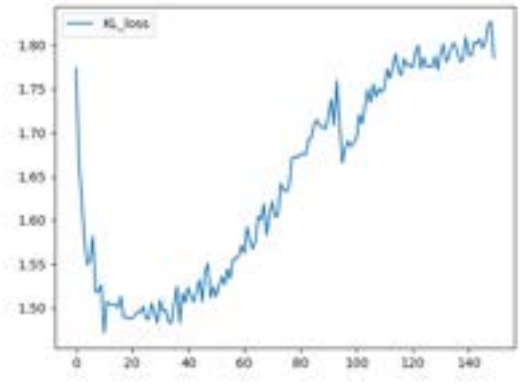
(a) Histogrammes des données générées par VAE-VP.



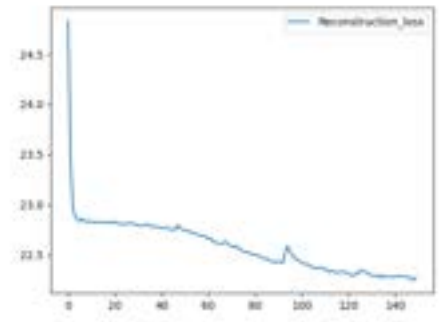
(b) Densité du VP sur l'espace latent.

FIGURE 2.8 – Caractéristique de l'apprentissage par VAE-VP avec la procédure de pré-entraînement

L'apprentissage est bien meilleur dans ce cas de figure, il y a une apparition de quatre modes bien distincts pour la distribution du VP. Cependant, certains modes sont mieux capturés que d'autres.



(a) Trajectoire de la perte de Kullback-Leibler pendant l'apprentissage du VAE-VP après pré-entraînement.



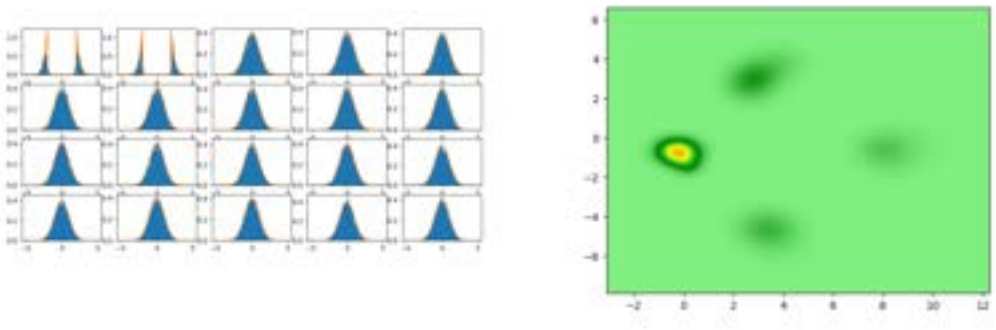
(b) Trajectoire de la perte de reconstruction pendant l'apprentissage du VAE-VP après pré-entraînement.

La perte D_{KL}, diminue, car elle cherche à coller au prior puis elle augmente dans un second temps. On évite le phénomène du posteriori-collapse. Pour la perte de reconstruction, elle décroît à

différente vitesse. Elle décroît plus lentement à partir du moment où la divergence de Kullback-Leibler commence à croître. Ceci illustre le compromis qu'il faut faire entre ces deux termes.

II.7 Apprentissage à l'aide du VAE Mélange de Gaussiennes

Pour ce prior, nous remarquons de moins bons résultats que pour le prior VP. Les performances concernant la réduction de dimension avec le MoG sont bien inférieures. En effet, il y a une répartition inégalitaire quant aux masses de probabilités dans les quatre zones d'intérêt.



(a) Histogrammes des données générées par VAE-MoG. **(b)** Densité du MoG sur l'espace latent

FIGURE 2.10 – Caractéristique de l'apprentissage par VAE-MoG avec la procédure de pré-entraînement (application EM)

De plus, il est assez souvent difficile d'obtenir une convergence de l'algorithme EM à partir des données encodées (et après initialisation de l'espace latent). On illustre ce cas dans la Figure 2.11. Ayant constaté la nécessité d'un bon pré-entraînement, on peut penser que les performances du VAE-MoG dans le cadre de notre étude, sont peu suffisantes.

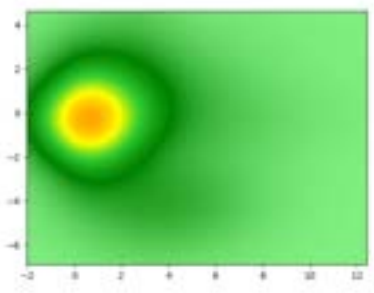


FIGURE 2.11 – Densité du prior MoG sur l'espace latent à la fin du pré-entraînement, soit après application de l'algorithme EM sur les données encodées dans l'espace latent.

Étant donné que le modèle VAE-VP a démontré une robustesse et une stabilité au fur à mesure des apprentissages, nous décidons de conserver uniquement celui-ci comme modèle génératif. Nous nous concentrons maintenant à utiliser le VAE-VP comme échantillonneur au sein de l'algorithme SS.

Chapitre 3

Algorithme VAE et SS

L'entraînement du VAE sur les données permet d'obtenir les paramètres $(\theta^*, \phi^*, \lambda^*)$ qui maximisent l'ELBO. Ces paramètres sont déterminés à l'aide d'une descente de gradients, avec des points d'initialisation définis par les procédures présentées dans la partie précédente. La question suivante est de savoir comment intégrer un modèle génératif de type VAE, dans l'algorithme SS décrit précédemment dans le chapitre 2.

I Subset Simulation et échantillonnage par VAE

Comme établi dans la Remarque 1, un bon noyau de transition correspond à la loi cible. En utilisant le VAE, nous pouvons apprendre et approximer les lois conditionnelles définies dans (1.8). Cela signifie que pour chaque densité $f_{X|F_j}$, nous l'approchons par un mélange infini de gaussiennes à l'aide du VAE :

$$f^{\theta^*}_{X|F_j}(x) = \int_{\mathcal{Z}} p^{\theta^*}_{X|F_j}(x \mid z) p_{\lambda^*}(z) dz \quad x \in \mathbb{R}^d. \tag{3.1}$$

Nous utilisons ensuite ce mélange infini comme noyau de proposition indépendant de l'observation précédente $x \in \mathbb{R}^d$, une observation provenant de la chaîne de Markov générée par l'algorithme M-H :

$$Q(. \mid x) = f^{\theta^*}_{X|F_j}(.) \quad \forall x \in \mathbb{R}^d. \tag{3.2}$$

Cela entraînerait l'apparition de l'expression suivante pour la probabilité d'acceptation :

$$\alpha(\tilde{x}, x_i) = \min\left\{1, \frac{f_{X|F_j}(\tilde{x}) f^{\theta^*}_{X|F_j}(x_i)}{f_{X|F_j}(x_i) f^{\theta^*}_{X|F_j}(\tilde{x})}\right\} \tag{3.3}$$

$$= \min\left\{1, \frac{f_X(\tilde{x}) f^{\theta^*}_{X|F_j}(x_i)}{f_X(x_i) f^{\theta^*}_{X|F_j}(\tilde{x})} \mathbf{1}_{F_j}(\tilde{x})\right\} \tag{3.4}$$

Malgré un processus d'échantillonnage simple

$$\begin{cases} z \sim p, & z \in \mathcal{Z} \\ x \sim p(. \mid z), & x \in \mathcal{X}, \end{cases}$$

l'expression de $f^{\theta^*}_{X|F_j}(x),\ x \in \mathbb{R}^d$ reste difficile à obtenir en raison de la nature infinie du mélange gaussien. Une option consiste donc à approximer la valeur de la densité et par conséquent de l'intégrale, en utilisant une méthode MCN :

$$\hat{f}^{\theta^*}_{X|F_j}(x) = \frac{1}{N_p} \sum_{n=1}^{N_p} p^{\theta^*}_{X|F_j}(x \mid Z^{(n)}) \quad \text{avec } (Z^{(n)})_n \overset{i.i.d}{\sim} p \tag{3.5}$$

Cependant, comme pour tout estimateur, il y a une erreur d'estimation et la valeur $\hat{f}^{\theta^*}_{X|F_j}(x)$ n'est donc

pas exacte. D'autant plus que nous avons déjà de l'incertitude, car $f^{\theta^*}_{X|F_j}$ est une approximation non paramétrique de f_X. Cela pose un problème pour la précision de l'algorithme M-H, où l'exactitude de la densité du noyau de proposition dans le ratio (3.3) est cruciale. En conséquence, une autre approche est nécessaire.

L'idée, dans ce manuscrit, est d'approcher le mélange infini de gaussienne $f^{\theta^*}_{X|F_j}$ par un mélange fini dont on peut calculer de manière exacte la densité en tout point $\mathbf{x} \in \mathbb{R}^d$. Voici la procédure suivie :

1. On génère $\{Z^{(1)}, \ldots, Z^{(N_p)}\}$, un N_p-échantillon selon le prior p_{λ^*}
2. On tire uniformément et avec remise N_p échantillons $\{\tilde{Z}^{(1)}, \ldots, \tilde{Z}^{(N_p)}\}$ dans $\{Z^{(1)}, \ldots, Z^{(N_p)}\}$ de telle sorte que $\{\tilde{Z}^{(1)}, \ldots, \tilde{Z}^{(N_p)}\}$ suit le processus empirique de p_{λ^*}, soit $p^{N_p}_{\lambda^*}$
3. Avec le décodeur D_{θ^*} du VAE, on génère les distributions gaussiennes associées $\left(p^{\theta^*}_{X|F_j}(. \mid \tilde{Z}^{(n)})\right)_n$, permettant de construire le mélange fini suivant :

$$Q(N_p)(.) = \frac{1}{N_p} \sum_{n=1}^{N_p} p^{\theta^*}_{X|F_j}(. \mid \tilde{Z}^{(n)}) \tag{3.6}$$

Ce mélange fini (3.6) devient alors le noyau de proposition dans l'algorithme M-H, remplaçant $f^{\theta^*}_{X|F_j}$ dans l'expression (3.3) par $Q(N_p)$.

Remark 6 Le paramètre N_p défini la taille du mélange gaussien. Il est choisi arbitrairement. Cependant, l'utilisateur doit garder à l'esprit qu'un entier N_p élevé impliquerait un coût computationnel pour l'évaluation de la densité $Q(N_p)$ en un point $x \in \mathbb{R}^d$ important.

Enfin, pour une amélioration des performances d'apprentissage du VAE de chaque densité $f_{X|F_j}$, nous choisissons de centrer et réduire les données d'apprentissage. Le principe repose sur la formulation suivante :

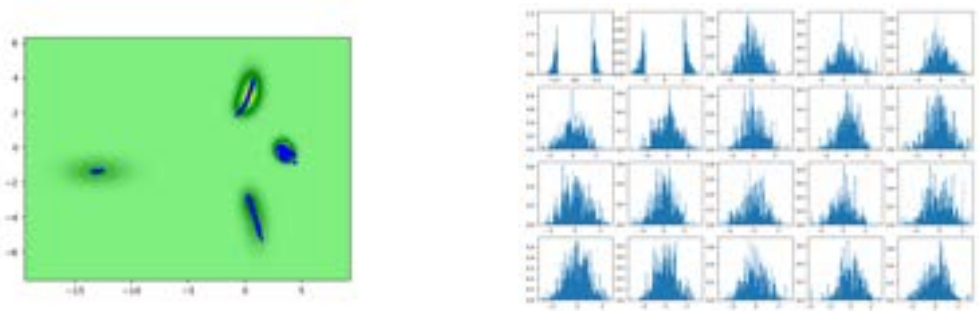
Soit $X \sim f_X$, $L = \text{diag}(\mathbb{V}ar(X))^{1/2}$, alors $U = L^{-1}(X - \mathbb{E}(X)) = G(X)$ est la variable centrée, réduite et sa densité a pour expression :

$$\begin{aligned} f_U(u) &= \mid det(G^{-1}(U)) \mid f_X(Lu + \mathbb{E}(X)) && (3.7)\\ &= \mid det(L) \mid f_X(Lu + \mathbb{E}(X)) && (3.8)\\ &= \prod_{i=1}^{d} (\mathbb{V}ar(X_i))^{1/2} \times f_X(Lu + \mathbb{E}(X)) && (3.9) \end{aligned}$$

Cette expression est obtenue à partir de la formule de changement de variable.

Avec ces outils en place, nous pouvons maintenant formuler l'algorithme impliquant l'échantillonnage par VAE, en tenant compte de l'effet du centrage et de la réduction des données (voir algorithme 3). Cette transformation conduit à une nouvelle expression de la probabilité d'acceptation (3.10). Le ratio de cette probabilité permet de faire disparaître le produit d'écart-type $\prod_{i=1}^{d} (\mathbb{V}ar(X_i))^{1/2}$.

Nous appliquons l'algorithme qu'on nomme *MCMC-VAE* à l'exemple des gaussiennes tronquées vu dans la section précédente. En résumé, on utilise la procédure d'échantillonnage expliquée ci-dessus dans un algorithme M-H classique et dont le noyau de proposition est $Q(N_p)$. Les points de la Figure 3.1 représentent l'encodage dans l'espace latent des valeurs de la chaîne de Markov. On observe qu'ils appartiennent tous aux quatre zones encodées. Enfin, les histogrammes montrent une similarité avec la vraie distribution, néanmoins, la qualité de reconstruction n'est pas complètement fidèle.



(a) Encodages des valeurs de la chaîne de Markov dans l'espace latent (points en bleu).

(b) Histogrammes des échantillons renvoyés par l'algorithme M-H.

FIGURE 3.1 – Application d'un algorithme M-H avec échantillonnage VAE-VP afin de simuler selon la densité de la troncature (2.23). On effectue une chaine de longueur $l = 1000$ et on choisit la taille du mélange gaussien $N_p = 300$.

Algorithm 3 Subset Simulation avec échantillonneur VAE indépendant

Require: — Seuil probabilité fixe p_0
— N-échantillon de départ $\{X^{(1)},\dots,X^{(N)}\}$
— $\{Y^{(1)},\dots,Y^{(N)}\} = \{\Phi(X^{(1)}),\dots,\Phi(X^{(N)})\}$
— Seuil limite s
— K pseudo-inputs
— N_p : nombre de gaussiennes pour le mélange fini

On pose $\overleftarrow{F}_N^{(k)}$ processus quantile de l'échantillon $E_{(k)} = \{Y_{(k)}^{(1)},\dots,Y_{(k)}^{(N)}\}$ et $\hat{\gamma}_N^{(k)} = \overleftarrow{F}_N^{(k)}(1-p_0)$ le seuil considéré au k^e événement et $A_{(k)} = \{X_{(k)}^{(1)},\dots,X_{(k)}^{(N)}\}$, l'échantillon associé.

$k = 1$

while $\hat{\gamma}_N^{(k)} < s$ **do**

On considère $\tilde{A}_{(k)} = \{X_{(k)}^{(i)}, \quad Y_{(k)}^{(i)} > \hat{\gamma}_N^{(k)} \ \forall i\}$, par définition de $\hat{\gamma}_N^{(k)}$, $card\left(\tilde{A}_{(k)}\right) = \lfloor Np_0 \rfloor$

$\tilde{U}_{(k)} = \{diag(\hat{\mathbb{V}}ar(\mathbf{X})) \times [X_{(k)}^{(i)} - \hat{\mathbb{E}}(\mathbf{X})], \quad X_{(k)}^{(i)} \in \tilde{A}_{(k)}\}$ Données centrées réduites

Apprentissage du VAE-VP sur l'échantillon $\tilde{U}_{(k)}$ pour définir le noyau de proposition :

1. Pseudo-inputs : $VP_{\lambda^*}(e_1\dots,e_K) = (u_1^{\lambda^*},\dots,u_K^{\lambda^*})$
2. $(\mu_k^{\lambda^*},\Sigma_k^{\lambda^*})_{k=1}^K = E_{\phi^*}(u_1^{\lambda^*},\dots,u_K^{\lambda^*})$ et on forme le prior VP p_{λ^*} définie dans (2.19).
3. $\{z^{(1)},\dots,z^{(N_p)}\}$ N_p-échantillon de p_{λ^*} pour définir le noyau de proposition $Q(N_p)(.)$ définie par (3.6).

On tire uniformément avec remise dans $\tilde{A}_{(k)}$ N échantillons : $A_{(k)}^* = \{X^{*\,(1)}_{(k+1)},\dots,X^{*\,(N)}_{(k+1)}\}$

On tire uniformément avec remise dans $\tilde{U}_{(k)}$ N échantillons : $U_{(k)}^* = \{U^{*\,(1)}_{(k+1)},\dots,U^{*\,(N)}_{(k+1)}\}$

Application de N algorithmes **M-H** dont chaque initialisation est un élément de $A_{(k)}^*$ et $U_{(k)}^*$.
On effectue des chaînes de longueur l dont la loi stationnaire est $f_{X|F_j}$.
La probabilité d'acceptation pour $\tilde{u} \sim Q(N_p)(.)$ est :

$$\min\left\{1, \frac{f_{X|F_j}\left(diag(\hat{\mathbb{V}ar}(\mathbf{X}))\tilde{u} + \hat{\mathbb{E}}(\mathbf{X})\right)Q(N_p)(u_i)}{f_{X|F_j}\left(diag(\hat{\mathbb{V}ar}(\mathbf{X}))u_i + \hat{\mathbb{E}}(\mathbf{X})\right)Q(N_p)(\tilde{u})}\right\} \tag{3.10}$$

$k = k+1$

On pose $A_{(k)} = \{X_{(k)}^{(1)}[l+1],\dots,X_{(k)}^{(N)}[l+1]\}$, $E_{(k)} = \{Y_{(k)}^{(1)}[l+1],\dots,Y_{(k)}^{(N)}[l+1]\}$ et $\hat{\gamma}_{(N)}^{(k)}$ associé

end while

return $P_f^{SSVAE} = p_0^{k-1}\dfrac{card\left\{Y_{(k)}^{(i)} > s,\ Y_{(k)}^{(i)} \in E_{(k)}\right\}}{N}$

II Application au cas 4-branches

Dans cette section, nous reprenons l'exemple test introduit dans le chapitre 1 et appliquons l'algorithme 3. Nous utilisons un échantillon de taille $N = 10\,000$. Ce choix est motivé par le fonctionnement de la méthode SS, qui ne conserve que $\lfloor Np_0 \rfloor$ échantillons à chaque étape. Ces échantillons constituent les données d'apprentissage de notre VAE. Pour assurer la robustesse de notre modèle, il est crucial de maintenir un nombre suffisant d'échantillons d'apprentissage, car une réduction excessive de ce nombre pourrait compromettre la qualité du VAE.

Contrairement au chapitre 1, pour l'étude de l'estimateur P_f^{SSVAE} , nous effectuons 50 estimations au lieu de 100. De plus, nous introduisons deux nouvelles quantités : la taille des chaînes (qui est variable ici) et le nombre d'estimations réussies. Nous discuterons de leur rôle et de leur importance dans la suite.

II.1 Application de l'algorithme SS pour le cas : $\beta = 3.5 \quad P_f = 9.3 \times 10^{-4}$

Dimension	$\hat{P_f}^{SSVAE}$	CV	ν	time	tailles chaînes	nombre d'estimations
$d = 5$	9.43×10^{-4}	0.043	1.34	$178s$	15	50 /50
$d = 10$	9.59×10^{-4}	0.043	0.98	$192s$	20	50 / 50
$d = 50$	9.56×10^{-4}	0.044	0.55	$277s$	35	47 /50
$d = 100$	9.53×10^{-4}	0.042	0.42	$1090s$	55	45/50

(a) Résumé de la qualité de l'estimateur P_f^{SSVAE}

Évènement	F_0	F_1	F_2	F_3	F_4
$d = 5$	0.36	0.28	0.24	0.21	0.19
$d = 10$	0.35	0.27	0.23	0.20	0.18
$d = 50$	0.34	0.25	0.21	0.19	0.18
$d = 100$	0.31	0.23	0.20	0.18	0.16

(b) Taux d'acceptation ρ pour chaque événement

TABLE 3.1 – Estimation par l'algorithme 3 pour l'événement où $\beta = 3.5$ et $P_f = 9.3 \times 10^{-4}$. La probabilité fixe $p_0 = 0.25$. Le noyau de proposition est le mélange fini de 300 gaussiennes issues du VAE. Pour les taux d'acceptation ρ, ce sont les taux moyennés sur l'ensemble des chaînes pour chaque événement F_i

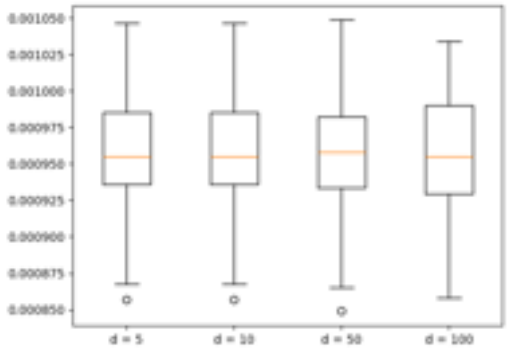
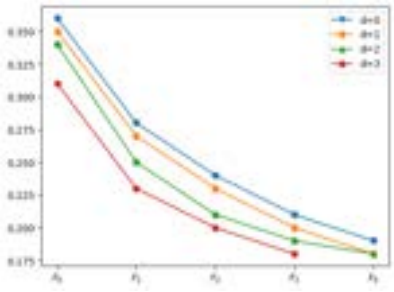


FIGURE 3.2 – Boite à moustache pour les différents estimateurs P_f^{SSVAE} correspondant à chaque dimension d pour le cas $\beta = 3.5$ et $P_f = 9.3 \times 10^{-4}$.

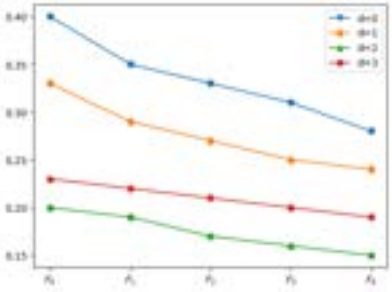
L'estimateur P_f^{VAESS} présente un CV autour des 4% Quelle que soit la dimension, ce qui témoigne d'une précision similaire à celle obtenue avec la méthode SS classique. Cependant, plusieurs limitations de performance ont été observées pour cet estimateur. Tout d'abord, les estimations produites sont biaisées. Bien que l'estimateur par méthode SS soit également biaisé, ce biais est généralement d'ordre $O(\frac{1}{N})$, donc relativement négligeable devant $N = 10\,000$. Ici, le biais semble plus important, et

une explication possible pourrait résider dans le temps nécessaire pour que les chaînes de Markov convergent vers leur état stationnaire. Pour pallier cela, nous avons tenté d'augmenter la taille des chaînes de Markov. Toutefois, étant donné que les biais deviennent plus importants dans les dimensions élevées (50 et 100), il a fallu ajuster la taille des chaînes en fonction de la dimension d. Cela pose problème, car une augmentation excessive de la taille des chaînes se traduit par un nombre accru d'appels au code Φ, et comme illustrer dans la Figure 3.2, les valeurs ν restent égales ou inférieures à 1, montrant ainsi que la méthode ne parvient pas à surpasser une approche MCN dans ce cas.

Un autre aspect crucial à considérer est le nombre d'estimations réussies sur l'ensemble des 50 tentatives effectuées. Une des difficultés rencontrées lors de la mise en œuvre de l'algorithme SSVAE réside dans l'échec de l'apprentissage du VAE à certaines étapes du processus SS. Travailler avec des modèles génératifs comporte le risque que l'optimisation échoue parfois, menant à des solutions non optimales. Nous avons observé des situations où les variances (Σ_z^θ), renvoyées par le décodeur D_θ étaient nulles ou quasi-nulles, ce qui rend impossible la construction de $Q(N_p)$ et oblige à interrompre l'algorithme.
Ce phénomène a été principalement constaté dans les dimensions 50 et 100, où 3 à 5 estimations ont rencontré des problèmes d'apprentissage du VAE. Cette difficulté est en grande partie liée à la malédiction de la dimensionnalité : dans des espaces de grande dimension, les points sont plus éloignés les uns des autres, ce qui conduit le VAE, malgré la régularisation, à attribuer aux points isolés une densité gaussienne avec une variance très faible.



(a) Résumé graphique du tableau des taux d'acceptation dans 3.1, soit avec l'algorithme SSVAE.



(b) Résumé graphique du tableau des taux d'acceptations dans 1.1, soit avec l'algorithme SS

FIGURE 3.3 – Évolution du taux d'acceptation moyen pour chaque dimension d en fonction de l'événement F_i considéré pour le cas $\beta = 3.5$ et $P_f = 9.3 \times 10^{-4}$.

En examinant le tableau des taux d'acceptation présentés dans la Figure 3.1, on observe une différence notable par rapport à celui de la Figure 1.1 qui résulte de l'application de l'algorithme SS 2. On résume ces deux tables dans la Figure 3.3. On constate qu'il y a une plus faible décroissance du taux ρ lorsqu'on augmente la dimension d du problème. Ces graphiques illustrent bien la robustesse du VAE à la grande dimension.

II.2 Application de l'algorithme SS pour le cas : $\beta = 5 \quad P_f = 1.15 \times 10^{-6}$

Pour le cas d'application $\beta = 5$ et $P_f = 1.15 \times 10^{-6}$, nous avons des constats similaires au cas d'application précédent, seulement les performances face à la méthode naïve MCN sont meilleures. Ceci n'est pas un résultat inattendu puisque la probabilité est de l'ordre de 10^{-6}. Une méthode SS doit-être plus performante. Toutefois, la dimension $d = 100$, semble poser plus de problèmes, on remarque un biais plus important ainsi que l'apparition d'outlier (voir Figure 3.4).

Dimension	$\hat{P_f}^{SSVAE}$	CV	v	time	tailles chaînes	nombre d'estimations
$d=5$	1.18×10^{-6}	0.060	310.34	$323s$	15	50 /50
$d=10$	1.18×10^{-6}	0.063	242.56	$347s$	20	50 / 50
$d=50$	1.27×10^{-6}	0.057	171.16	$504s$	35	49 /50
$d=100$	1.42×10^{-6}	0.164	14.46	$1966s$	55	43/50

(a) Résumé de la qualité de l'estimateur P_f^{SS}

Évènement	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
$d=5$	0.36	0.28	0.24	0.21	0.20	0.19	0.17	0.16	0.15
$d=10$	0.35	0.27	0.23	0.20	0.18	0.16	0.16	0.15	0.13
$d=50$	0.34	0.25	0.21	0.19	0.18	0.16	0.15	0.14	0.14
$d=100$	0.31	0.23	0.20	0.18	0.16	0.15	0.14	0.13	0.13

(b) Taux d'acceptation ρ pour chaque événement

TABLE 3.2 – Estimation par l'algorithme 3 pour l'événement où $\beta=5$ et $P_f=1.15\times10^{-6}$. La probabilité fixe $p_0=0.25$. Le noyau de proposition est le mélange fini de 300 gaussiennes issues du VAE. Pour les taux d'acceptation ρ, ce sont les taux moyennés sur l'ensemble des chaînes pour chaque événement F_i

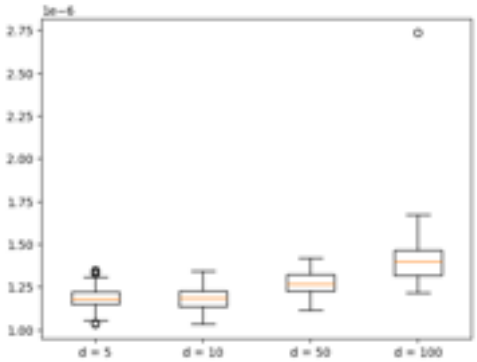
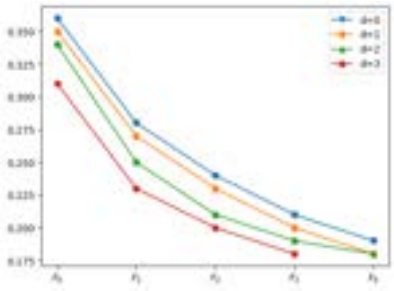
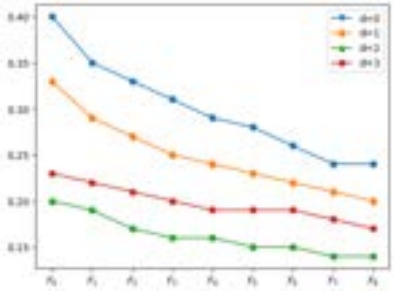


FIGURE 3.4 – Boite à moustache pour les différents estimateurs P_f^{SSVAE} correspondant à chaque dimension d pour le cas $\beta=5$ et $P_f=1.15\times10^{-6}$.



(a) Résumé graphique du tableau des taux d'acceptation dans 3.1.



(b) Résumé graphique du tableau des taux d'acceptations dans 1.1.

FIGURE 3.5 – Évolution du taux d'acceptation moyen pour chaque dimension d en fonction de l'événement F_i considéré pour

Nous explorons le comportement de notre algorithme en examinant de près la forme des densités, en particulier celle des priors. Étant définies sur $\mathbb{R}^2$, il est plus simple de les visualiser. On ajoute parfois à cette représentation l'encodage de différents points :

1. En bleu, l'encodage des échantillons d'apprentissage.
2. En violet, l'encodage des échantillons renvoyés par l'algorithme M-H

Pour commencer, examinons le cas de la dimension 5. À cette dimension, aucun problème d'apprentissage notable n'est observé. Au fur et à mesure que les événements progressent, la Figure 3.6 montre clairement l'apparition de quatre zones distinctes, certaines ayant des masses de probabilité plus élevées que d'autres. Toutefois, lorsqu'on observe la Figure avec les différents points 3.7, un décalage entre les échantillons d'apprentissage et ceux issus des chaînes de Markov devient apparent.

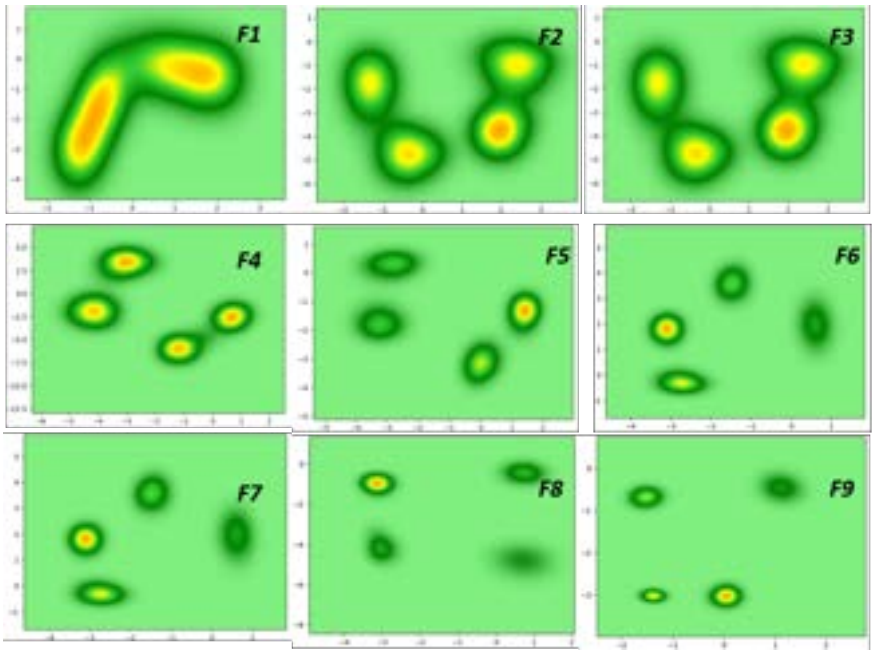


FIGURE 3.6 – Densité VP apprise à chaque évènement F_j du SS lorsque les données d'apprentissage vivent dans l'espace $\mathbb{R}^5$

Il est important de se rappeler que seule la densité $f_{X|F_1}$ est approximée par le VAE avec un échantillon distribué selon la bonne loi, i.e $f_{X|F_1}$. Les autres distributions ont des approximations basées sur des échantillons issus des précédentes chaînes de Markov. Par conséquent, on assiste à une propagation des erreurs sur les différentes estimations $f^{\theta^*}_{X|F_j}$, et par conséquent à une dégradation progressive du VAE.

Lorsqu'on examine de plus près le comportement de chaque chaîne à chaque étape de l'événement, on observe qu'au sein de certaines, aucune proposition n'est acceptée. Toutefois, pour passer des $\lfloor Np_0 \rfloor$ échantillons qui satisfont l'appartenance à F_j (voir ligne 7 de l'algorithme 3), on fait du bootstrap, ceci induit l'apparition de doublons. Cet échantillon, avec doublons, deviendra l'entrée des algorithmes M-H (voir ligne 9 de l'algorithme 3). Par conséquent, s'il y a un taux d'acceptation de 0 pour certaines chaînes, ceci signifie que nous avons plus de chance de retrouver des doublons au sein des données d'apprentissage du VAE suivant.

Nous avons fait également quelques observations sur les comportements des chaînes. Par exemple, pour cette dimension, à l'événement F_1, nous avons 118 chaînes qui rejettent toutes les propositions. Parmi, ces 118 échantillons qui n'ont pas bougé, on retrouve huit valeurs de vecteurs qui ont une occurrence supérieure ou égale à 2. Puis au dernier événement F_9, 1410 échantillons n'ont pas bougé et parmi ces échantillons 335 valeurs de vecteur ont une occurrence supérieure ou égale à 2.

On retrouve également ces shift des données encodées pour les autres dimensions (voir annexe II.1).

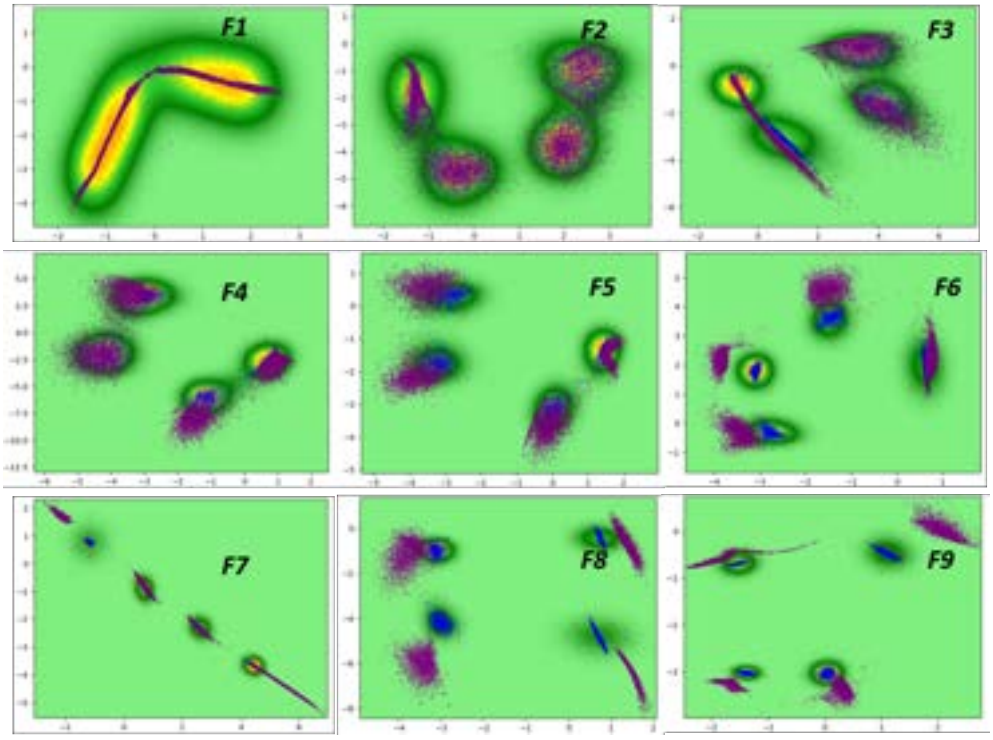


FIGURE 3.7 – Densité VP apprise à chaque évènement F_j du SS lorsque les données d'apprentissage vivent dans l'espace $\mathbb{R}^5$

Pour revenir à notre réflexion sur l'estimateur en dimension $d = 100$ au début de la section II.2, il est important de souligner que cette dimension pose de sérieux problèmes d'apprentissage dans le cadre de l'algorithme VAE-SS. Comme le montre la figure 10, le VAE peine à identifier les quatre modes, en particulier pour l'événement F_4, qui illustre bien cette difficulté. On retrouve en annexe

les graphiques des marginales des lois conditionnelles $(f_{X|F_j})_j$, pour le cas de la dimension $d = 5$. On trace les marginales renvoyées par l'apprentissage VAE et on compare à l'aide d'une méthode MCN. Il est important de noter que ces graphiques sont seulement tracés à titre indicatif, les densités $(f_{X|F_j})_j$ ne définissent pas des distributions dont les composantes sont indépendantes en raison de la présence de la présence de l'indicatrice $\mathbf{1}_{F_j}$ qui empêche cela.

Conclusion

Dans ce travail, nous nous sommes intéressés à la méthode des SS et à ses implications, en particulier dans le contexte de l'utilisation des algorithmes MCMC. Bien que cette méthode soit puissante, elle présente plusieurs défis en grande dimension, notamment ceux liés à la qualité des chaînes générées. Des questions se posent naturellement quant à savoir si le régime stationnaire a été atteint et à partir de quel moment les échantillons peuvent être considérés comme bien distribués selon la loi cible. De plus, explorer efficacement l'espace des états reste une tâche complexe, surtout lorsque nous avons une connaissance limitée des formes des distributions $f_{X|F_j}$.

Pour répondre à ces défis, nous avons proposé une méthode d'estimation des probabilités de défaillance par SS en intégrant un modèle génératif, le VAE. L'un des objectifs principaux était d'augmenter les taux d'acceptation au sein de l'algorithme. Cependant, cette tâche s'est révélée extrêmement complexe avec l'introduction du VAE. Malgré cela, nous avons réussi à ralentir la décroissance des taux d'acceptation avec l'augmentation de la dimension d, ce qui représente une amélioration notable.

L'introduction du VAE avait pour but de capturer la complexité des distributions, en particulier lorsque celles-ci présentent des structures de dépendance complexes. Cependant, dans le cadre des SS, la dégradation progressive de la qualité des échantillons au fur et à mesure de l'avancement dans les événements $(F_j)_j$ a entravé l'efficacité de l'apprentissage du VAE.

Il serait pertinent d'explorer de nouvelles méthodes pour améliorer le processus de récupération des données d'apprentissage en amont du modèle VAE. Une perspective intéressante pourrait consister à utiliser des échantillons pondérés, ce qui pourrait potentiellement améliorer la robustesse et l'efficacité de l'apprentissage, en particulier dans des contextes de grande dimension. La méthode Sequential Importance Sampling (SIS) peut être un point de départ de réflexion. Il s'agirait de remplacer $\mathbf{1}_{F_j}$ par une fonction "lisse", par exemple la fonction de répartition d'une gaussienne [12].

Enfin, nous concluons ce rapport en abordant certaines difficultés techniques rencontrées lors de ce stage. L'une des principales difficultés concernait la mise en œuvre des modèles VAE sur TensorFlow. En effet, l'utilisation de la classe **Model** a entraîné des fuites de mémoire. Grâce au package *memory-profiler*, nous avons pu surveiller l'état de la mémoire à chaque étape de notre algorithme et avons constaté que la mémoire occupée par un modèle VAE n'était jamais libérée. Cette fuite de mémoire provoque une consommation progressive des ressources, rendant l'exécution de l'algorithme de plus en plus lourde.

Cette situation a rendu l'obtention de propriétés fiables sur l'estimateur P_f^{SSVAE} plus complexe que prévu. Effectuer plus de 15 estimations sur une machine ordinaire s'est avéré impraticable. En conséquence, nous avons dû recourir à un supercalculateur de développement pour mener à bien nos simulations et obtenir les résultats souhaités.

I Preuves théoriques

Cette section contient les preuves de certaines assertions citées dans le rapport

I.1 Probabilité de défaillance pour l'exemple 4-branches

Comme introduit dans l'équation (1.11), on donne le détail du calcul pour l'approximation de la probabilité de défaillance :

$$\begin{aligned}
P_f = \mathbb{P}(\Phi(\mathbf{X}) \geq 0) &= \mathbb{P}(-\min\{\beta \pm t_i, \quad i = 1,2\} \geq 0) \\
&= \mathbb{P}(\max\{\pm t_i, \quad i = 1,2\} \geq \beta) \\
&= \mathbb{P}(\max\{|\ t_1\ |, |\ t_2\ |\} \geq \beta) \\
&= 1 - \mathbb{P}(\max\{|\ t_1\ |, |\ t_2\ |\} \leq \beta) \qquad (11) \\
&\overset{(\text{cov}=0)}{=} 1 - \mathbb{P}(|\ t_1\ | \leq \beta)^2 \qquad (12) \\
&= 1 - (1 - 2\phi_{\mathcal{N}(0,1)}(-\beta))^2 \\
&= 4\phi_{\mathcal{N}(0,1)}(-\beta) - 4\phi_{\mathcal{N}(0,1)}(-\beta)^2 \\
&\approx 4\phi_{\mathcal{N}(0,1)}(-\beta)
\end{aligned}$$

Le passage de la ligne (11) à (12) est possible car en calculant la covariance entre t_1 et t_2, on s'aperçoit que celle-ci est nulle

$$\begin{aligned}
\text{cov}(t_1, t_2) &= \frac{1}{d}\left[\text{cov}\left(\sum_{i=1}^{d} x_i, \sum_{i=1}^{\lfloor d/2 \rfloor} x_i\right) - \text{cov}\left(\sum_{i=1}^{d} x_i, \sum_{i=1+\lfloor d/2 \rfloor}^{d} x_i\right)\right] \\
&= \frac{1}{d}\left[\lfloor d/2 \rfloor - \lfloor d/2 \rfloor\right] = 0
\end{aligned}$$

Et de même pour $\text{cov}(t_1, -t_2) = \text{cov}(-t_1, t_2) = \text{cov}(-t_1, -t_2) = 0$.

I.2 Calcul de l'entropie dans le cas Gaussien

On suppose que l'espace latent $\mathcal{Z}$ est $\mathbb{R}^{d_z}$. On a pour tout $z \in \mathbb{R}^{d_z}$, l'expression de l'opposé de l'entropie définie dans 2.9.

$$\begin{aligned}
E_{q_\phi(\cdot|x)}[\log q_\phi(Z \mid x)] &= \int \log\Big(q_\phi(z \mid x)\Big) q_\phi(z \mid x)\, dz \\
&= \int \frac{\exp\left[-\frac{1}{2}\sum_{i=1}^{d_z} \frac{\left(z_i - \mu_x^\phi(i)\right)^2}{\left(\Sigma_x^\phi\right)_{ii}}\right]}{(2\pi)^{d_z/2}\left(\prod_{i=1}^{d_z}\left(\Sigma_x^\phi\right)_{ii}\right)^{1/2}} \times \left[-\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{d_z}\log\left(\left(\Sigma_x^\phi\right)_{ii}\right) + -\frac{1}{2}\sum_{i=1}^{d_z}\frac{\left(z_i - \mu_x^\phi(i)\right)^2}{\left(\Sigma_x^\phi\right)_{ii}}\right] dz \\
&= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{d_z}\log\left(\left(\Sigma_x^\phi\right)_{ii}\right) - \frac{1}{2}\sum_{i=1}^{d_z}\int \frac{\left(z_i - \mu_x^\phi(i)\right)^2}{\left(\Sigma_x^\phi\right)_{ii}} \times \frac{\exp\left[-\frac{1}{2}\sum_{i=1}^{d_z} \frac{\left(z_i - \mu_x^\phi(i)\right)^2}{\left(\Sigma_x^\phi\right)_{ii}}\right]}{(2\pi)^{d_z/2}\left(\prod_{i=1}^{d_z}\left(\Sigma_x^\phi\right)_{ii}\right)^{1/2}} dz \\
&\overset{\text{Fubini positif}}{=} -\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{d_z}\log\left(\left(\Sigma_x^\phi\right)_{ii}\right) - \frac{1}{2}\sum_{i=1}^{d_z}\underbrace{\mathbb{V}(T)}_{=1} \quad \text{avec } T \sim \mathcal{N}(0,1) \\
&= -\frac{1}{2}\sum_{i=1}^{d_z}\log\left[2\pi\left(\Sigma_x^\phi\right)_{ii} e\right]
\end{aligned}$$

II Graphiques supplémentaires

II.1 Graphiques des espaces latents lors de l'application de l'algorithme VAE-SS

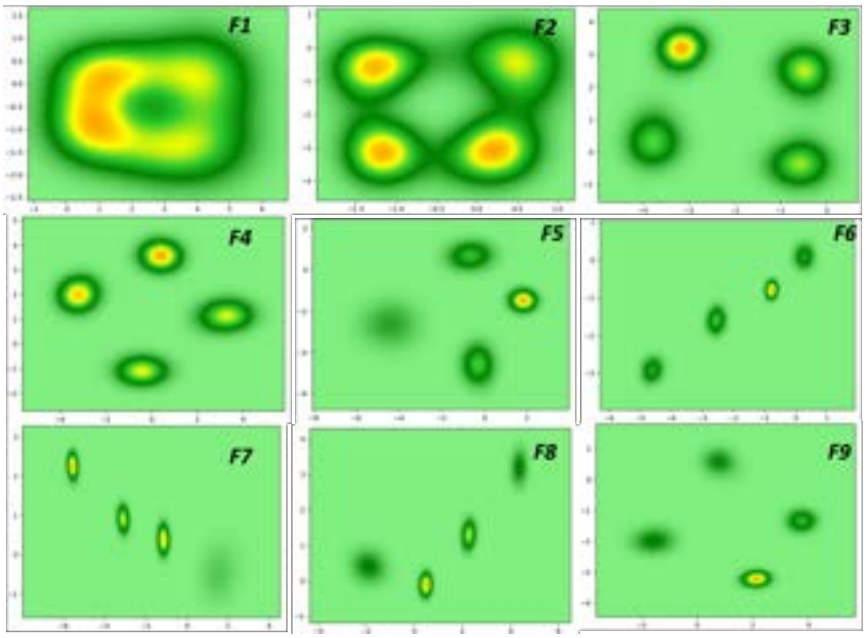


FIGURE 8 – Densité VP apprise à chaque évènement F_j du SS lorsque les données d'apprentissage vivent dans l'espace $\mathbb{R}^{10}$

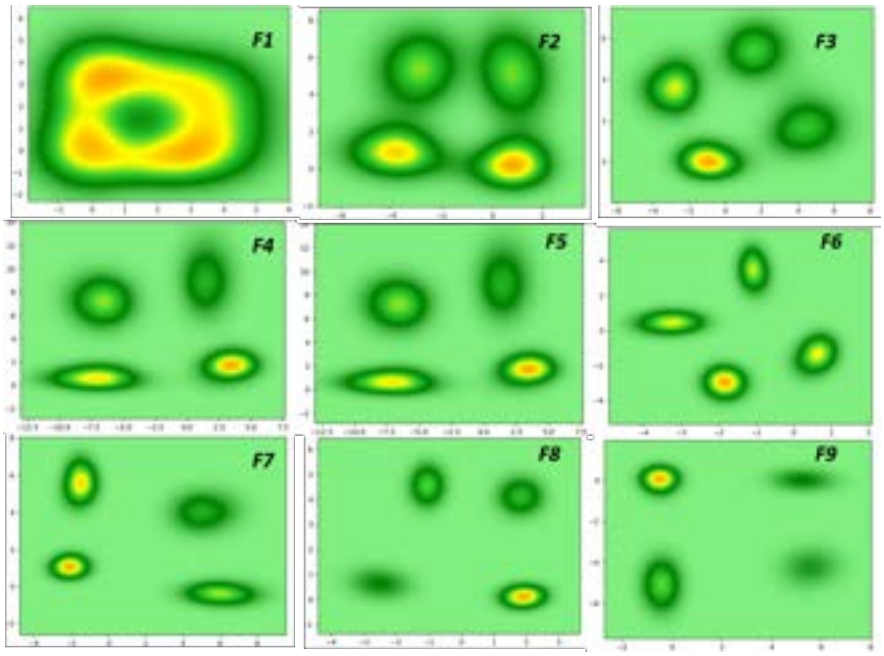


FIGURE 9 – Densité VP apprise à chaque évènement F_j du SS lorsque les données d'apprentissage vivent dans l'espace $\mathbb{R}^{50}$

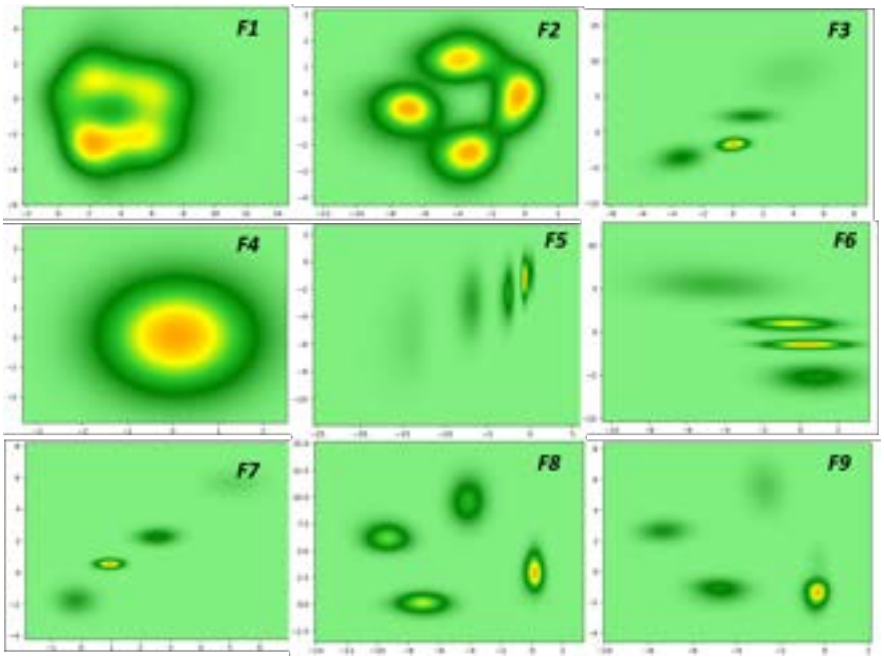


FIGURE 10 – Densité VP apprise à chaque évènement F_j du SS lorsque les données d'apprentissage vivent dans l'espace $\mathbb{R}^{100}$

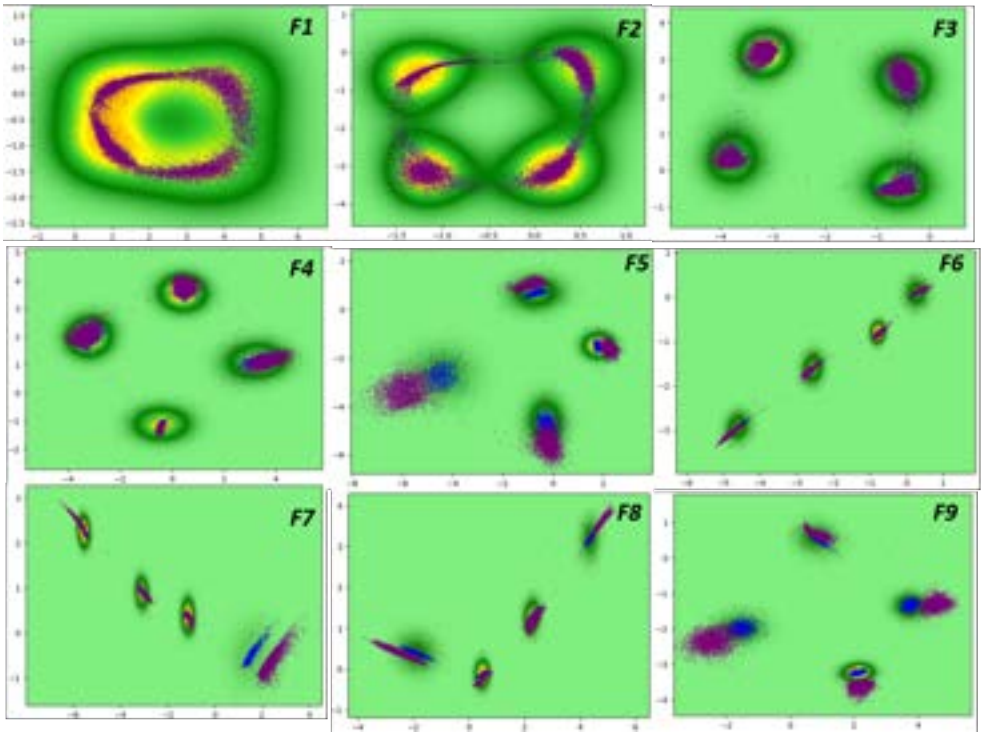


FIGURE 11 – Densité VP apprise à chaque évènement F_j du SS lorsque les données d'apprentissage vivent dans l'espace $\mathbb{R}^{10}$. Les points bleus correspondent aux données d'apprentissages encodées et ceux en violet aux échantillons issus des chaînes de Markov encodées.

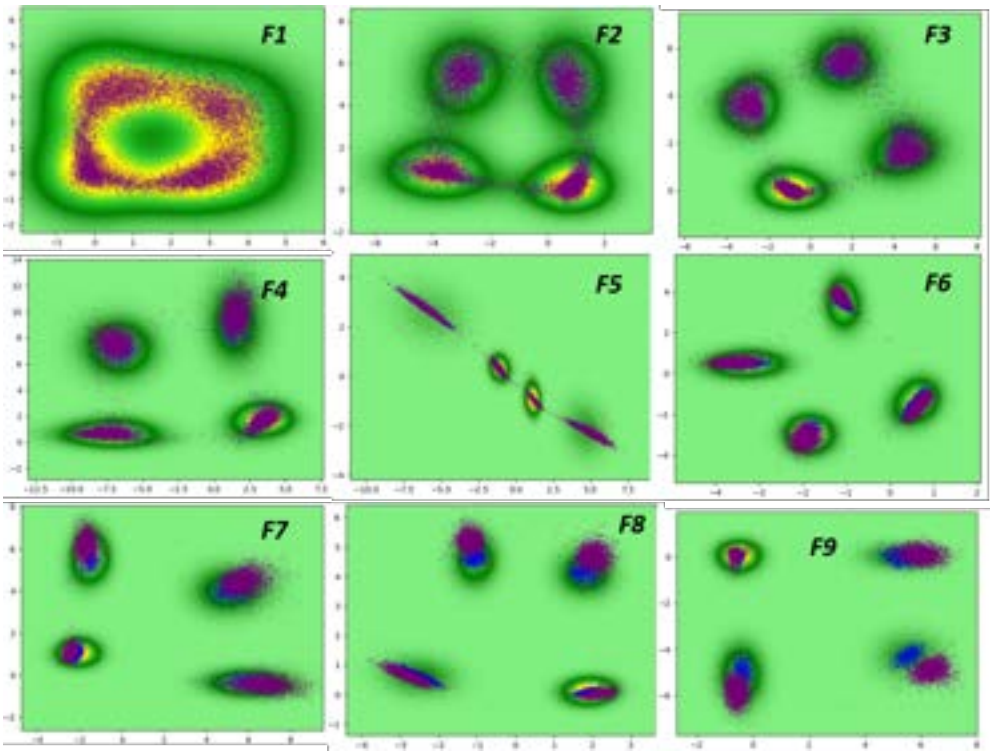


FIGURE 12 – Densité VP apprise à chaque évènement F_j du SS lorsque les données d'apprentissage vivent dans l'espace $\mathbb{R}^{50}$. Les points bleus correspondent aux données d'apprentissages encodées et ceux en violet aux échantillons issus des chaînes de Markov encodées.

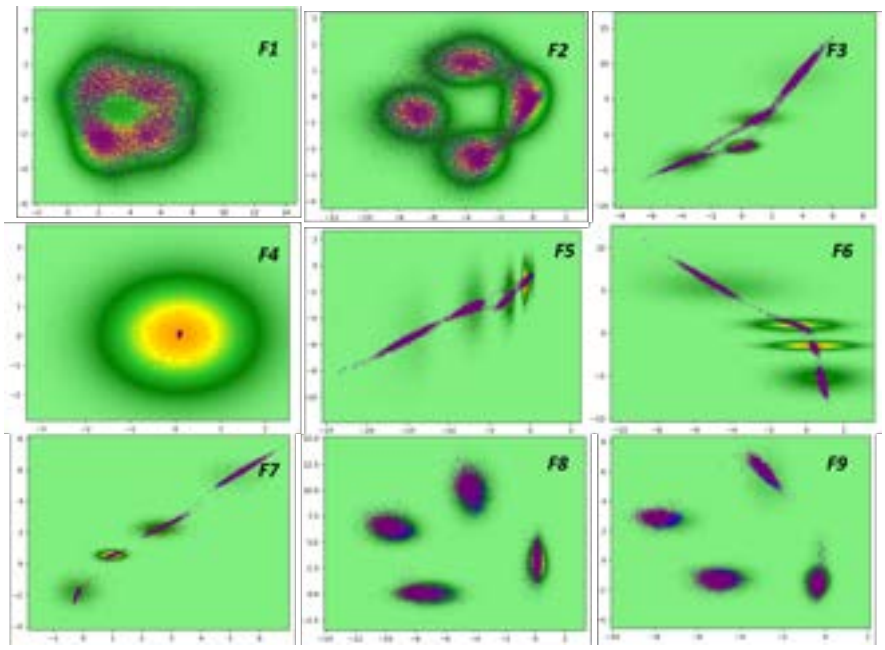


FIGURE 13 – Densité VP apprise à chaque évènement F_j du SS lorsque les données d'apprentissage vivent dans l'espace $\mathbb{R}^{100}$. Les points bleus correspondent aux données d'apprentissages encodées et ceux en violet aux échantillons issus des chaînes de Markov encodées.

II.2 Graphiques des marginales lors de l'application de l'algorithme VAE-SS

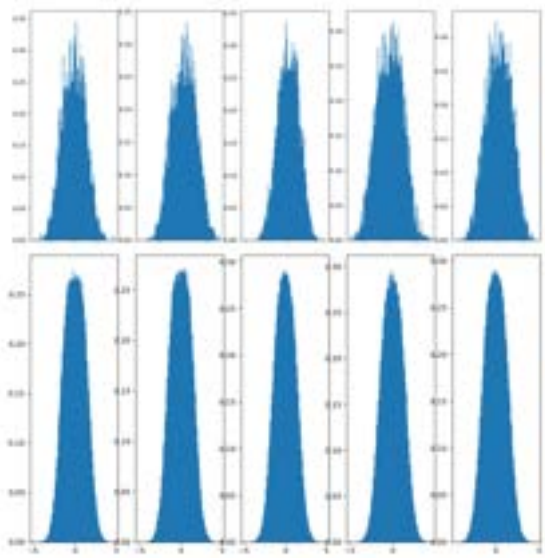


FIGURE 14 – Histogramme des marginales pour $f_{X|F_1}$, en haut les histogrammes basés sur la distribution apprise par le VAE à la première étape de l'algorithme SS, en bas une approximation des marginales par méthode MCN.

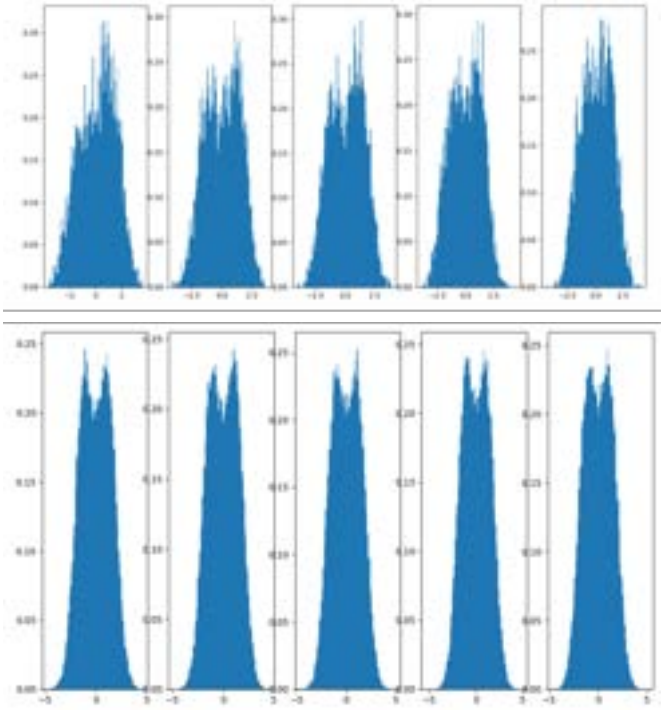


FIGURE 15 – Histogramme des marginales pour $f_{X|F_2}$, en haut les histogrammes basés sur la distribution apprise par le VAE à la deuxième étape de l'algorithme SS, en bas une approximation des marginales par méthode MCN.

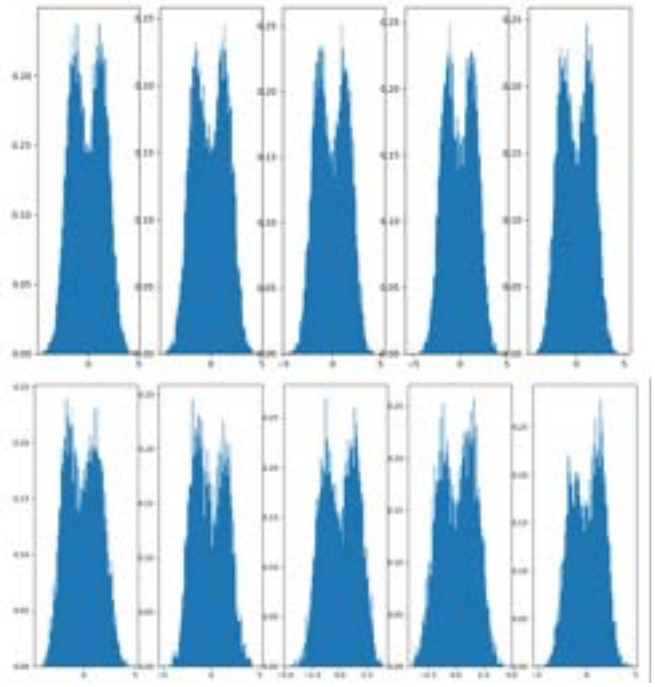


FIGURE 16 – Histogramme des marginales pour $f_{X|F_3}$, en haut les histogrammes basés sur la distribution apprise par le VAE à la troisième étape de l'algorithme SS, en bas une approximation des marginales par méthode MCN.

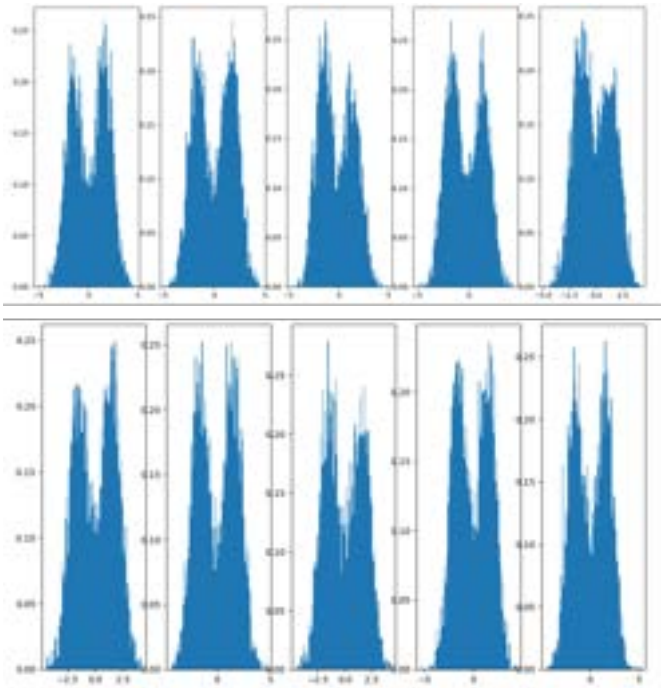


FIGURE 17 – Histogramme des marginales pour $f_{X|F_4}$, en haut les histogrammes basés sur la distribution apprise par le VAE à la quatrième étape de l'algorithme SS, en bas une approximation des marginales par méthode MCN.

Bibliographie

[1] Siu-Kui Au and James L. Beck. Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic Engineering Mechanics*, 16(4) :263–277, October 2001.

[2] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space, May 2016. arXiv :1511.06349 [cs].

[3] Frédéric Cérou, Pierre Del Moral, Teddy Furon, and Arnaud Guyader. Sequential monte carlo for rare event estimation. *Statistics and computing*, 22(3) :795–808, 2012.

[4] Julien Demange-Chryst, François Bachoc, Jérôme Morio, and Timothé Krauth. Variational autoencoder with weighted samples for high- dimensional non-parametric adaptive importance sampling.

[5] Matthew D Hoffman and Matthew J Johnson. ELBO surgery : yet another way to carve up the variational evidence lower bound.

[6] Herman Kahn and Theodore E Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series*, 12 :27–30, 1951.

[7] Diederik P. Kingma and Max Welling. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4) :307–392, 2019. arXiv :1906.02691 [cs, stat].

[8] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders, May 2016. arXiv :1511.05644 [cs].

[9] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4) :115–133, December 1943.

[10] Jérôme Morio, Mathieu Balesdent, Damien Jacquemart, and Christelle Vergé. A survey of rare event simulation methods for static input–output models. *Simulation Modelling Practice and Theory*, 49 :287–304, December 2014.

[11] Iason Papaioannou, Wolfgang Betz, Kilian Zwirglmaier, and Daniel Straub. MCMC algorithms for Subset Simulation. *Probabilistic Engineering Mechanics*, 41 :89–103, 2015.

[12] Iason Papaioannou, Costas Papadimitriou, and Daniel Straub. Sequential importance sampling for structural reliability analysis. *Structural Safety*, 62 :66–75, September 2016.

[13] Christian P. Robert. The Metropolis-Hastings algorithm. January 2016. arXiv :1504.01896 [stat].

[14] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B : Statistical Methodology*, 61(3) :611–622, 1999.

[15] Jakub M. Tomczak and Max Welling. VAE with a VampPrior, February 2018. arXiv :1705.07120 [cs, stat].

[16] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *Advances in neural information processing systems*, 29, 2016.

[17] Konstantin Zuev. Subset Simulation Method for Rare Event Estimation : An Introduction, May 2015. arXiv :1505.03506 [stat].